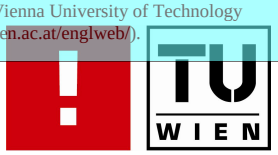

FAKULTÄT FÜR !NFORMATIK

Semantische Abfragen zur Dynamischen Inhaltserstellung

Diplomarbeit

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Medieninformatik

eingereicht von

Christoph Wastyn
Matrikelnummer 0100878

an der:
Fakultät für Informatik der Technischen Universität Wien

Betreuung:
Betreuerin: Univ.Ass. Dipl.-Ing. Dr.techn. Monika Lanzenberger

Wien, 12. Oktober 2008

(Unterschrift Verfasser) (Unterschrift Betreuerin)

Kurzfassung

Die Organisation von Information durch sogenannte Content Management Systeme gewinnt im Internet durch die große Zahl an Inhalten zunehmend an Bedeutung. Durch eine Trennung von Inhalt und Layout werden RedakteurInnen entlastet, da sich ein System um die technische Umsetzung der Struktur einer Website kümmert. Während in den letzten Jahren viele der technischen Entwicklungen des Internet in solchen Systemen Einzug fanden, sind semantische Dienste in diesem Zusammenhang noch weitgehend unbekannt. Nach dem Konzept des semantischen Internet werden Begriffe miteinander in Beziehung gesetzt und zu einem großen Wissensnetz zusammengefügt, innerhalb dessen dezentral vorliegende Zusammenhänge erkannt werden können. Würde eine Seite etwa Tokio als Hauptstadt deklarieren, und eine andere Quelle als Stadt Japans, so würde sich daraus der Zusammenhang ableiten lassen, dass Tokio die Hauptstadt Japans sei. Die semantische Vernetzung von Content Management Systemen würde nicht nur den Zugriff dezentral vorhandener Information ermöglichen, im Sinne des redaktionellen Prozesses könnten diese Informationen auch verwendet werden, um semantische Inhalte semiautomatisch, und damit wesentlich zeitsparender, zu publizieren. Obwohl viele Arbeiten auf das große Potential semantischer Vernetzung von Content Management Systemen verweisen, gibt es derzeit nur wenige praktische Anwendungen, die den Einsatz dieser Technologie ermöglichen.

Diese Arbeit enthält eine Einführung in das semantische Internet im Zusammenhang mit Content Management Systemen, um die Grundlage für ein Verständnis beider Konzepte zu schaffen. Es wird darauf eingegangen, welche Auswirkungen das semantische Internet auf die Organisation und Verarbeitung von Information im Internet haben könnte, und eine Analyse aktueller Content Management Systeme soll zeigen, in wie weit diese bereits heute für das Semantic Web gerüstet sind. Im Anschluss wird anhand eines praktischen Beispiels gezeigt, wie semiautomatische Inhaltserstellung in CMS genutzt werden kann. Zu diesem Zweck wird ein Plugin für eine Internetplattform entwickelt, das die Erstellung von Webseiten auf Basis semantischer Abfragen ermöglicht. Die Arbeit endet mit einer Zusammenfassung und Diskussion der einzelnen Konzepte und einem Ausblick in zukünftige Entwicklungen, die das Bild des Internet auf Dauer verändern könnten.

Inhaltsverzeichnis

1 Einleitung **1**
- 1.1 Motivation . . . 1
- 1.2 Zielsetzung . . . 2
- 1.3 Aufbau der Arbeit . . . 2

2 Eine Einführung in das Semantic Web **5**
- 2.1 Die Semantic Web Vision . . . 6
- 2.2 Von Wissensrepräsentationen zum Semantic Web . . . 9
- 2.3 Technologien des Semantic Web . . . 10
  - 2.3.1 HTML, XML und DTD . . . 11
  - 2.3.2 RDF und RDFS . . . 14
  - 2.3.3 Web Ontology Language . . . 17
  - 2.3.4 Semantische Abfragesprachen . . . 19
- 2.4 Anwendungen des Semantic Web . . . 19

3 Semantic Web und Content Management **25**
- 3.1 Grundlagen . . . 26
- 3.2 Vom statischen Web zum Web Content Management . . . 27
- 3.3 Web Content Management und Web 2.0 . . . 29
- 3.4 Semantic Web und Content Management . . . 30
- 3.5 Semantic Web Standards . . . 32
  - 3.5.1 SIOC . . . 33
  - 3.5.2 Triplify . . . 34
  - 3.5.3 Individuelle Ontologien . . . 34
- 3.6 Stolpersteine auf dem Weg zum Semantic Web Content Management . . . 35
- 3.7 Forschungsfragen . . . 36
- 3.8 Analyse der Systeme . . . 37
  - 3.8.1 Typo 3 . . . 39
  - 3.8.2 Joomla! . . . 41
  - 3.8.3 Drupal . . . 42

3.8.4 MODx . . . 45
3.8.5 WordPress . . . 47
3.9 Diskussion . . . 49

4 Praktischer Teil - Grundlagen 50
4.1 Einleitung . . . 50
4.2 Entwicklung in Drupal . . . 51
4.2.1 Nodes . . . 53
4.2.2 Blöcke und Boxen . . . 54
4.2.3 Module und Hooks . . . 54
4.3 Die RDF API . . . 58
4.3.1 RDF API Konzepte . . . 59
4.3.2 RDF Funktionen . . . 61
4.3.3 RDF API Hooks . . . 61
4.4 DBpedia . . . 61
4.4.1 Datenstruktur . . . 62
4.4.2 Datenzugriff . . . 64
4.5 SPARQL API . . . 64
4.5.1 SPARQL API Klassen . . . 64
4.5.2 SPARQL API Funktionen . . . 65
4.5.3 SPARQL API Hooks . . . 65

5 Praktischer Teil - Umsetzung 66
5.1 Planung . . . 66
5.1.1 Anforderungsdefinition . . . 66
5.1.2 Vorgehensmodell . . . 68
5.2 Anforderungsanalyse . . . 68
5.2.1 Anwendungsfälle . . . 69
5.2.2 Szenarien . . . 72
5.3 Entwurf . . . 76
5.3.1 Systemdesign . . . 76
5.3.2 Datenbank . . . 79
5.3.3 Mock-up . . . 79
5.4 Implementierung . . . 82
5.4.1 Installation . . . 82
5.4.2 Erstellung des Node Typs . . . 83
5.5 Programmtests . . . 88
5.5.1 Setting . . . 88

5.5.2 Ergebnisse 90
5.5.3 Feedback und Diskussion 91

6 Abschluss **93**
6.1 Zusammenfassung 93
6.2 Diskussion 94
6.2.1 Reflexion aus Entwicklungssicht 95
6.3 Ausblick 96

7 Danksagung **97**

8 Literaturverzeichnis **98**

9 Anhänge **104**
9.1 Anhang A: Hooks in Drupal 104
9.2 Anhang B: birthday.module Datei 107

1 Einleitung

In diesem Kapitel soll die Motivation für die Arbeit herausgestrichen werden. Ebenso wird die daraus resultierende Zielsetzung beschrieben, die den Zweck und die Aufgaben der Arbeit definiert. Im Anschluss wird der Aufbau der Arbeit näher beschrieben, um eine allgemeine Orientierung zu schaffen.

1.1 Motivation

Die Idee für eine Arbeit im Bereich semantischer Content Management Systeme (CMS) entstand bei der Erstellung einer Website, die Inhalte aus der öffentlichen Enzyklopädie Wikipedia[1] verwendete, um Information darzustellen. Der recht umständliche Prozess diese Inhalte manuell zu übernehmen und zu formatieren, führte zu der Frage, ob es eine automatisierte Methode gibt, verschiedene Inhalte zu extrahieren, um sie in einem Content Management System zu integrieren. Das Konzept des Semantic Web sieht einen solchen dezentralen Zugriff auf externe Daten vor, die über Abfragesprachen wie SPARQL[2] zusammengestellt werden können. Obwohl es im Internet viele Beispielanwendungen des Semantic Web, und mit DBpedia[3] sogar eine semantische Darstellung von Wikipedia gibt, schien kein Content Management System eine geeignete Schnittstelle für semantische Dienste zur Verfügung zu stellen.

Aus diesem Grund stellte sich die Frage nach den Möglichkeiten solcher Schnittstellen, und ob es bereits Bestrebungen in CMS gibt, diese zu nutzen. Nachdem erste Recherchen ergaben, dass einige Arbeiten zwar ein hohes Potential semantischer Vernetzung von CMS sehen, es jedoch nur vereinzelte Anwendungen gibt, entstand die Idee, eine Schnittstelle zu DBpedia zu implementieren, über die semantische Inhalte in solchen Systemen integrierbar werden sollen.

[1] http://www.wikipedia.org (Stand: 01.10.08)
[2] http://www.w3.org/TR/rdf-sparql-query/ (Stand: 01.10.08)
[3] http://dbpedia.org (Stand: 01.10.08)

1.2 Zielsetzung

Aus dieser Motivation lässt sich nun die Zielsetzung der Arbeit ableiten, die sich wie folgt definieren lässt:

- Es soll ein tieferes Verständnis für das Potential des Semantic Web in Bezug auf das Content Management gewonnen werden.
- Bestehende Systeme sollen auf die Verfügbarkeit semantischer Technologien hin untersucht werden.
- Auf Basis der Ergebnisse soll eine Schnittstelle implementiert werden, die eine Integration von semantischen Daten in ein Content Management System ermöglicht.

Um dieses Ziel zu erreichen, muss aber erst das nötige Grundwissen erarbeitet werden. Dies umfasst sowohl eine Darstellung der Konzepte, als auch eine Analyse einzelner Content Management Systeme, um zu ermitteln, inwieweit das technologische Gerüst für eine semantische Schnittstelle bereits vorhanden ist. Nach dieser theoretischen Analyse soll eine praktische Implementierung eines Moduls erfolgen, dass eine semantische Schnittstelle zu der Datenbank DBpedia ermöglicht. Hierzu müssen ein geeignetes System auf Basis der Forschungsfragen ausgewählt, und die Grundlagen für eine Arbeit mit diesem System geschaffen werden. Der Aufbau der Arbeit und der einzelnen Kapitel wird im nächsten Abschnitt beschrieben.

1.3 Aufbau der Arbeit

Die Arbeit gliedert sich in einen theoretischen und einen praktischen Teil. Im theoretischen Teil sollen die wesentlichsten Aspekte des semantischen Internet und des Content Management vorgestellt werden. Ebenso umfasst dieser Teil eine Darstellung semantischer Abfragesprachen und bestehender Konzepte, um CMS zu vernetzen. Es folgt ein Querschnitt durch die wichtigsten CMS, um die Verfügbarkeit semantischer Technologien zu bewerten. Im praktischen Teil wird das technische Grundgerüst eines auf Basis der Analyse ausgewählten CMS näher beleuchtet und der Entwurf einer semantischen Schnittstelle dieses Systems zu DBpedia erstellt. Dieser Entwurf soll im Rahmen eines Vorgehensmodells umgesetzt und bewertet werden. Zuletzt werden die Ergebnisse der Arbeit diskutiert. Die folgende Aufstellung enthält eine Kurzdarstellung aller Kapitel der Arbeit:

Kapitel 2: Einführung in das Semantic Web Dieses Kapitel enthält einen Überblick über die Vision und Motivation des semantischen Internet. Zunächst werden die Idee und die möglichen Anwendungsfelder des Semantic Web betrachtet und der Vergleich zu Systemen zur Wissensrepräsentation gezogen. Danach folgt eine genaue Darstellung der einzelnen Technologien des Semantic Web. Dies umfasst die Internet Auszeichnungssprachen HTML[4] und XML[5], die semantische Beschreibungssprache RDF[6] und darauf aufbauende Sprachen. Ebenso wird die Abfragesprache SPARQL vorgestellt. Das Kapitel endet mit einigen Anwendungsbeispielen, die das Potential des Semantic Web verdeutlichen.

Kapitel 3: Semantic Web und Content Management Im nächsten Kapitel werden die Grundlagen zum Verständnis von Content Management Systemen im Zusammenhang mit dem Semantic Web erläutert. Nach einer Einführung in die wichtigsten Elemente des Content Management wird insbesondere die Bedeutung einer semantischen Vernetzung derartiger Systeme beleuchtet. Ebenso wird auf bestehende Möglichkeiten zur Vernetzung von CMS durch externe und systemeigene Schnittstellen eingegangen. Zuletzt folgt eine systematische Analyse der wichtigsten CMS, um die Verfügbarkeit der technologischen Voraussetzungen einer solchen semantischen Vernetzung zu ergründen.

Kapitel 4: Praktischer Teil - Grundlagen In diesem Kapitel wird der praktische Teil der Arbeit skizziert. Zunächst wird das Content Management System Drupal[7] vorgestellt, das als Basis für die Implementierung einer semantischen Schnittstelle dient. Neben einer Einführung in die Entwicklung in diesem System, werden die zur Verfügung stehenden Programmierschnittstellen beleuchtet, die im Zuge der Implementierung verwendet werden. Ebenso wird das technologische Gerüst der semantischen Datenbank DBpedia besprochen, die als Quelle für semantische Abfragen genutzt werden soll.

Kapitel 5: Praktischer Teil - Umsetzung Die Umsetzung der semantischen Schnittstelle wird im nächsten Kapitel beschrieben. Die Gliederung richtet sich nach dem Vorgehensmodell und besteht aus einer Planungsphase, einer Anforderungsanalyse, einem Entwurf, der Implementierung und BenutzerInnentest. Das Kapitel beschreibt die Funktionalität und den Umfang der Implementierung, enthält die Definition eines geeigneten Modells, verschiedene Anwendungsszenarien und die

[4] http://www.w3.org/TR/1999/REC-html401-19991224/ (Stand: 15.09.08)
[5] http://www.w3.org/TR/2006/REC-xml-20060816/ (Stand: 15.09.08)
[6] http://www.w3.org/RDF/(Stand: 01.10.08)
[7] http://drupal.org (Stand: 01.10.08)

Programmarchitektur des Moduls. Zuletzt wird das fertige Programm in BenutzerInnentests anhand der Szenarien untersucht und aus Anwendungssicht bewertet.

Kapitel 6: Abschluss Das abschließende Kapitel besteht aus einer Diskussion der erzielten Ergebnisse, in der die wichtigsten Erkenntnisse zusammengefasst werden. Danach folgen eine Reflexion der Implementierung aus Entwicklungssicht und eine Zusammenfassung der Kernelemente der Arbeit. Das Kapitel endet mit einem Ausblick auf mögliche zukünftige Entwicklungen im Bereich des semantischen Web Content Management.

2 Eine Einführung in das Semantic Web

Internetseiten bieten unterschiedlichste Inhalte, reichend von kommerziellen Angeboten bis hin zu öffentlichen Interessensgemeinschaften. Suchmaschinen und CMS tragen eine entscheidende Rolle bei der Suche und Darstellung von Daten in der enormen Vielfalt an Information und zählen zu den selbstverständlichsten Eckpfeilern des Internet von heute. Jedoch sind sie nicht in der Lage, die Bedeutung von Inhalten zu verarbeiten, da keine semantische Beschreibung dieser Daten existiert. Die im Internet gebräuchliche Auszeichnungssprache Hypertext Markup Language (HTML)[1] wurde in Hinblick auf die Lesbarkeit und für eine einfache Anwendung konzipiert. Sie enthält zwar Anweisungen wie ihre Daten im Webbrowser darzustellen sind, die inhaltlichen Informationen wird vom System jedoch nicht interpretiert. Eine semantische Beschreibung dieser Daten würde es Programmen ermöglichen, Seiten nicht nur nach Schlagworten, sondern nach semantischen Beschreibungen der Inhalte, zu durchsuchen. Information die über mehrere Seiten verstreut vorliegt, könnte zu einem Ganzen zusammengetragen und auf BenutzerInnen zugeschnitten werden. In weiterer Konsequenz könnten Software Agenten eines Tages unterschiedlichste Aufgaben mit den im Internet vorliegenden Daten erfüllen können, etwa die Planung von Terminen unter Berücksichtigung aktueller Internetdaten oder die Berechnung einer geeigneten Reiseroute auf Basis von Verkehrs- und Staumeldungen.

Dieses Kapitel gibt eine kurze Einführung in die Vision des semantischen Internet und welche Auswirkungen es für zukünftige Anwendungen haben könnte. Es folgt eine Darstellung des allgemeinen Konzeptes des Semantic Web und dessen Umsetzung im Internet durch die Auszeichnungssprache Extensible Markup Language (XML)[2] den Ressource Description Framework (RDF)[3], sowie das Resource Description Framework Schema

[1] http://www.w3.org/TR/1999/REC-html401-19991224/ (Stand: 15.09.08)
[2] http://www.w3.org/TR/2006/REC-xml-20060816/ (Stand: 15.09.08)
[3] http://www.w3.org/RDF/ (Stand: 15.09.08)

(RDFS)[4] und die darauf aufbauende Web Ontology Language (OWL)[5]. In diesem Zusammenhang werden der Aufbau von einfachen Internetseiten und die möglichen Hürden einer Verbreitung des Konzeptes erläutert. Zuletzt wird beleuchtet, in wie weit das Semantic Web heute bereits im Internet verbreitet ist und welche Anwendungen semantischer Verarbeitung von Daten bestehen.

2.1 Die Semantic Web Vision

Das Internet von heute ist das Produkt zahlloser Angebote, die in erster Linie von und für BenutzerInnen geschaffen wurden, sei es um Produkte zu verkaufen, gemeinschaftlich zu arbeiten, oder mit Interessensgruppen zu diskutieren. Dokumente und Informationen sind so aufbereitet, dass sie von Menschen leicht gelesen werden können. Selbst die Auszeichnungssprachen zur Erstellung von Inhalten sind für eine schnelle und unkomplizierte Verwendung konzipiert. Jedoch sind die vorhandenen Werkzeuge nur bedingt geeignet, um diese auf Menschen gerichtete Art der Benutzung zu unterstützen. Suchmaschinen haben heute eine selbstverständliche Rolle im Internet eingenommen und dienen zur Suche und zur Strukturierung der großen Masse an Angeboten. Anstatt aber eine gezielte Antwort auf eine Suchabfrage zu liefern, kann eine Suchmaschine nur eine gewertete Liste an Internetadressen ausgeben, die die gesuchte Information enthalten könnten. Diese Reihenfolge richtet sich lediglich nach der Popularität der Seite, der Anzahl der Verweise und ähnlichen Gesichtspunkten. Eine Suchmaschine kann die Relevanz oder Bedeutung eines Dokumentes nicht beurteilen und liefert somit höchst unpräzise Ergebnisse [Antoniou, 2004]. Aus diesem Grund werden Metadaten und Schlagworte in Seiten oft nur platziert, um in der Reihung von Suchmaschinen nach oben zu klettern, unabhängig davon ob die vorliegende Seite damit exakt beschrieben wird. Das macht es umso schwieriger, Internetseiten nach ihrer Relevanz zu ordnen [Diestelkamp, 2005].

Selbst wenn die Ergebnisse nach einer gut gefilterten Suche vorliegen, müssen diese erst gesichtet und erneut von BenutzerInnen gefiltert werden. Wenn beispielsweise eine Suchabfrage viele Treffer liefert, ist es nahezu unmöglich wichtige Informationen aus der Unzahl an Seiten herauszulesen. Wenn die gesuchte Information über unterschiedliche Seiten verstreut vorliegt, muss diese durch mehrere Suchabfragen zusammengetragen werden. Eine Internetsuche bleibt also eine äußerst unpräzise und zeitaufwändige Angelegenheit. In kleineren Organisationssystemen, sogenannten Content Management Sys-

[4] http://www.w3.org/TR/rdf-schema/ (Stand: 15.09.08)
[5] http://www.w3.org/TR/owl-features/ (Stand: 15.09.08)

temen, die den Inhalt eines einzigen, meist umfangreichen Webauftrittes organisieren, ist es oft einfacher eine gezielte Information zu finden, jedoch bleiben die Probleme auch in diesem Rahmen bestehen. Zwar erlauben kleine CMS die Strukturierung von Information nach den Vorgaben der ErstellerInnen, es ist aber nicht möglich Inhalte nach persönlichen Kriterien zu organisieren. Auch die Integration von externen Inhalten gestaltet sich aufgrund der schwierigen maschinellen Lesbarkeit von Internetdokumenten schwierig.

Tim Berners-Lee, der die noch heute gültige Internet Auszeichnungssprache HTML begründete, veröffentlichte 2001 einen Artikel [Berners-Lee, 2001], in dem er von einem Internet träumte, das die semantische Bedeutung der dargebotenen Information verstehen und verarbeiten könnte. Nach diesem Konzept werden Begriffe miteinander in Beziehung gesetzt und zu einem großen Wissensnetz zusammengefügt, innerhalb dessen dezentral vorliegende Zusammenhänge erkannt werden können. Würde eine Seite etwa *Tokio* als Hauptstadt deklarieren, und eine andere Quelle als Stadt Japans, so würde sich daraus der Zusammenhang ableiten lassen, dass Tokio die Hauptstadt Japans sei. In einem derart beschaffenen Wissensnetz würden sich die genannten Problemfelder durch eine maschinelle Auswertung der Daten beheben lassen. So könnten Suchabfragen semantisch ausgewertet werden und natursprachlich gestellte Fragen korrekt beantworten können. Auch für andere Anwendungsfälle ergeben sich durch das Semantic Web zahlreiche absehbare Vorzüge. Grigoris Antoniou [Antoniou, 2004] fasst diese Anwendungsfelder zusammen:

Knowledge Management: Als Knowledge Management bezeichnet man die Verwaltung und den Zugriff auf Wissen innerhalb einer Organisation durch ein Organisationssystem. Dieses Wissen kann verstreut vorliegen und durch ein entsprechendes Knowledge Management System gesucht, extrahiert, betrachtet oder durch implizite Zusammenhänge gewonnen werden.

- Durch das Semantic Web könnte es erleichtert werden, die Erhaltung von Wissen durch aufgespürte Inkonsistenzen zu erleichtern, oder neues Wissen aus nicht offensichtlichen Zusammenhängen zu gewinnen.
- Stichwort basierte Suchen könnten durch gezielte Abfragen ersetzt werden.
- Wissen könnte auch aus mehreren Quellen zusammengetragen werden.
- Ein Rechtesystem könnte den Zugriff auf Information nach Benutzergruppen verwalten.

B2C Handelsbeziehungen: In Handelsbeziehungen zwischen Unternehmen und Privatpersonen steht das Internet vor dem Problem, dass ähnliche Angebote von unzähligen Anbietern vertrieben werden und stichwortbasierte Suchen den direkten Vergleich zwischen der Qualität des Shops und den Preisen nur bedingt zulassen. Aus diesem Grund existieren zahlreiche Plattformen, die On Line Shops mittels Wrapper vergleichen indem Verkaufsdaten extrahiert und in einer Datenbank gespeichert werden.

- Mit dem Semantic Web können Mechanismen entwickelt werden, die den Export und die korrekte Verarbeitung dieser Information nach individuellen Bedürfnissen erlauben.
- Information über die Qualität eines Shops können dezentral aus unabhängigen Wertungssystemen abgeleitet werden.
- Das Programmieren von Wrappern wird durch die semantische Interpretation der vorliegenden Daten ersetzt.

B2B Beziehungen: Ein weiteres mögliches Anwendungsfeld des Semantic Web liegt im Kommunikations- und Handelsbereich zwischen Unternehmen. Das Konzept könnte die Möglichkeit eröffnen, den aufwändigen Ansatz des elektronischen Datenaustausches zwischen Unternehmen (EDI) durch semantische Beschreibungen von Daten über das Internet relativ einfach und kostengünstig abzuwickeln, und direkt in andere Anwendungen zu integrieren.

Persönliche Agenten: In einem zukünftigen Szenario könnten Software Agenten mit verteilten Internetdaten komplexe Aufgaben, wie die Planung von Terminen unter Berücksichtigung von Öffnungszeiten, Verkehrsmeldungen oder ähnlichen Aspekten, übernehmen. In diesem Zusammenhang sind der Fantasie kaum Grenzen gesetzt, da jegliche im Internet vorliegende Information in unterschiedlichsten Anforderungen zum Erreichen des Ziels verwendet werden könnte. Denkbare Bereiche wären etwa Reiseplanungen, Terminverwaltung, Einkauf, Produktauswahl, Kommunikation, usw.

Berners-Lee nannte dieses Internet, das die Bedeutung seiner Inhalte verstehen würde, *Semantic Web* und legte den Grundstein für eine Entwicklung, die in den letzten Jahren langsam aber stetig fortschritt und womöglich das größte Potential für eine Veränderung des Internets der letzten Jahre mit sich bringt.

2.2 Von Wissensrepräsentationen zum Semantic Web

Das Semantic Web folgt in den Grundzügen dem Prinzip von Wissensrepräsentationen, obgleich derartige Systeme heute noch wenig in der Praxis verbreitet sind. Umso mehr könnte das Semantic Web ein globales Anwendungsfeld für die Umsetzung dieser Idee darstellen [Berners-Lee, 2001]. In jedem Fall kann das Konzept von Wissensrepräsentationen herangezogen werden, um die Grundlagen semantischer Beschreibung zu beleuchten. Wissensplattformen verwenden zentral gespeicherte Daten, die klaren Konventionen und Definitionen folgen. Dies setzt ein einheitliches Vokabular und eine klar definierte Syntax voraus. Es wird dann nach den definierten Regeln möglich, nach bestimmten Informationen zu suchen, oder diese aus unterschiedlichen Zusammenhängen abzuleiten. Um Daten semantisch zu beschreiben, bedient man sich hierbei Taxonomien oder Ontologien.

Taxonomien dienen zur Klassifikation von Gegenständen, die auch als Entitäten bezeichnet werden. Es werden Klassen von Objekten definiert, die jeweils genau eine übergeordnete Klasse besitzen. Die Klasse *Tokio* könnte die Oberklasse *Japan* besitzen, und die Klasse *Japan* ihrerseits die Oberklasse *Länder Asiens*. Ausgehend von einem Wurzelknoten entsteht so eine Baumstruktur, die nach Unten hin immer mehr Details zu ihren übergeordneten Knoten besitzt. Eine solche Baumstruktur eröffnet also eine einfache Semantik, jedoch unter einigen Einschränkungen. Es kann schwierig werden, neue Begriffe in einen bestehenden Baum zu integrieren. Ebenso erlaubt es eine Baumstruktur nicht, Begriffe mehreren Oberklassen zuzuordnen, obwohl dies manchmal erforderlich wäre [Diestelkamp, 2005]. So kann *Japan* neben der Oberklasse *Länder Asiens* auch durchaus die Oberklassen *Mitgliedsstaat der UNO* und *Austragungsorte der olympischen Spiele* besitzen. Die Konsequenz daraus ist, statt einer hierarchischen Baumstruktur ein Netz zu verwenden, in dem Klassen durch logische Beziehungen verknüpft sind.

Ein solches Netz wird als Ontologie bezeichnet. Sie beschreibt eine spezifische Wissensdomäne formal und besteht aus einer endlichen Liste von Begriffen und den Beziehungen zwischen ihnen [Antoniou, 2004]. Einerseits können in einer Ontologie hierarchische Beziehungen durch Subklassen definiert werden, wobei diese nicht der monohierarchischen Einschränkung einer Taxonomie unterliegen. Andererseits erlaubt eine Ontologie, Eigenschaften, Werteeinschränkungen und logischen Relationen zu spezifizieren. Dadurch wird ein gemeinsames Vokabular für einen Wissensbereich festgelegt, für das Begriffe und Relationen klar definiert sind und entsprechend miteinander ausgetauscht werden können. Es muss klar geregelt sein, was - bleibt man bei dem genannten Beispiel - unter einem

Land verstanden wird. Andernfalls könnte es von einer Seite als *Staat*, von der anderen Seite als *Ackerboden* verstanden werden. Die Ontologie gewährleistet, dass Information einheitlich beschrieben wird, und erlaubt damit auch die Interpretation dieser Information durch maschinelle Mechanismen. Ähnlich wie in einem Organisationssystem kann eine Ontologie zur Repräsentation und Wiederverwendung von Web Content verwendet werden und die Kommunikation von Daten ermöglichen [Beck, 2004]. Im Internet können derartige Wissensbeziehungen durch Vokabular Beschreibungssprachen wie RDFS und OWL realisiert werden. Die folgenden Abschnitte widmen sich nun der Beschreibung dieser Technologien, und welche Voraussetzungen für ihre Anwendung nötig sind.

2.3 Technologien des Semantic Web

Internetseiten werden durch Auszeichnungssprachen wie HTML beschrieben. Darin können sowohl Texte als auch Angaben zum Layout der Seite enthalten sein. Jedoch werden die inhaltlichen Informationen nicht näher in maschineninterpretierbarer Form beschrieben, daher ist keine semantische Verarbeitung des Inhaltes möglich. Aus diesem Grund benötigt das semantische Web eine Auszeichnungssprache, in der Inhalte mittels einer standardisierten Syntax beschrieben werden. Dies kann durch die Metasprache XML geschehen, die zur Kommunikation zwischen IT Systemen dient. XML liefert eine einheitliche Syntax für den Datenaustausch, jedoch enthält sie keinerlei Angaben zur Struktur eines Dokuments. Um das gemeinsame Vokabular für einen Datenaustausch zu schaffen, existieren diverse Schemensprachen wie die Dokumenttypdefinition (DTD)[6] oder XML Schema[7], in denen Begriffe eindeutig definiert werden können. Durch die Verwendung einer Schemensprache wird ein XML-Dokument gültig, wenn es wohlgeformt ist und das definierte Format einhält. Als wohlgeformt bezeichnet man das Dokument, wenn es den durch XML spezifizierten Regeln folgt:

- Das Dokument besitzt genau ein Wurzelelement.
- Alle Elemente mit Inhalt besitzen eine Beginn- und eine Endkennung (tags).
- Die Beginn- und Endkennungen (tags) sind ebenentreu paarig verschachtelt.
- Ein Element darf nicht mehrere Attribute mit demselben Namen besitzen.

Somit weiß jedes System, welche Begriffe bei der Erstellung eines Dokuments verwendet werden dürfen. Um auch die Bedeutung dieser Begriffe festzulegen benutzt das Semantic

[6] http://edition-w3c.de/TR/2000/REC-xml-20001006/ (Stand: 15.09.08)
[7] http://www.edition-w3c.de/TR/2001/REC-xmlschema-0-20010502/ (Stand: 15.09.08)

Web den Ressource Description Framework, der als Auszeichnungssprache für Metadaten dient um semantische Beziehungen darzustellen. Diese werden als XML konforme Dokumente erzeugt und nutzen eine standardisierte Syntax wie die XML Syntax oder die Notation 3 (N3)[8]. Das Vokabular eines RDF Dokuments kann durch die Schemasprache RDFS festgelegt werden. Diese enthält keinerlei Vorgaben an zu verwendende Begriffe, sondern bietet nur den Rahmen um eigene Schemen festzulegen. Da jedoch für einen sinnvollen Datenaustausch Begriffe einheitlich definiert und verstanden werden müssen haben sich Schemengemeinschaften zusammengeschlossen, die versuchen einheitliche Vokabulare zu entwickeln, beispielsweise der Dublin Core [9]. Die Web Ontology Language setzt auf RDFS auf und bietet mehr Freiheiten bei der Erstellung einer Ontologie. So können beispielsweise Klassen durch boolesche Kombinationen verbunden oder flexible Werteeinschränkungen getroffen werden. Um die Komplexität je nach Anwendungsfall gering zu halten, existiert OWL auf drei Sprachebenen (OWL Lite, OWL DL und OWL Full). Um spezielle Informationen aus nun semantisch vorliegenden Daten zu gewinnen, gibt es darüber hinaus semantische Abfragesprachen wie die SPARQL Protocol and RDF Query Language (SPARQL)[10]. Mit ihnen können Datenbestände getrennt und nach beliebigen Gesichtspunkten gefiltert werden. Im Folgenden werden die Komponenten des Semantic Web im Einzelnen näher vorgestellt.

2.3.1 HTML, XML und DTD

Internetseiten bestehen aus unterschiedlichen Inhalten wie Texten, Bilder, Animationen oder kleinen Programmen. All diese Elemente besitzen eindeutige Adressen (URIs) über die sie angesteuert werden können. Die Struktur und das Layout einer solchen Seite werden in einer sogenannten Auszeichnungssprache festgelegt, die aber nicht nur Anweisungen für das Layout, sondern auch direkt den darzustellenden Text enthält. Die ursprüngliche und noch heute im Internet gebräuchliche Auszeichnungssprache ist die Hypertext Markup Language (HTML)[11], die anfang der 90er Jahre von Tim Berners-Lee entwickelt wurde [Raggett, 1999]. Durch eine Auszeichnung von Textteilen mittels *tags*, die maschinelle Information zur Formatierung enthalten, wird dem Dokument eine gewisse Struktur verliehen. Dabei wird ein Tag von einem Paar spitzer Klammern gekennzeichnet. Diese Klammern enthalten die Anweisung, wie der eingeschlossene Textteil zu verarbeiten ist. Beispielsweise enthält der HTML Tag *Beispieltext* die

[8] http://www.w3.org/DesignIssues/Notation3.html (Stand: 15.09.08
[9] http://dublincore.org/ (Stand: 29.06.08)
[10] http://www.w3.org/TR/rdf-sparql-query/ (Stand: 15.09.08)
[11] http://www.w3.org/TR/2006/REC-xml-20060816/ (Stand: 15.09.08)

Anweisung, dass der Inhalt zwischen den Klammern in einer fettgedruckten Schriftart ausgegeben werden soll. Obwohl die meisten Tags am Ende der Anweisung geschlossen werden, existieren auch offene Tags, wie *<hr>*, was eine horizontale Linie erzeugt. Daneben erlauben Tags auch die Einbindung von Tabellen, Listen oder externen Dateien wie Bilder oder Videoclips. Der HTML Ausschnitt in Listing 2.1 erzeugt einen formatierten Text, in dem Tokio als Hauptstadt Japans beschrieben wird.

Listing 2.1: HTML Code Beispiel

```
<html>
  <body>
    <b>Länder Asiens</b>
    <hr><p>
    Japan hat die Hauptstadt Tokio
  </body>
<html>
```

Das HTML Dokument wird durch das Tag Paar *<html></html>* eingeschlossen, der eigentliche Inhalt steht im *<body></body>* Teil. Daneben enthält der Ausschnitt eine horizontale Trennlinie und einen Absatz, der durch den Tag *<p>* spezifiziert wird. Obwohl dieses Beispiel leicht gelesen werden kann, ist der eigentliche Inhalt maschinell schwer zu interpretieren, da keine Annotation zur Bedeutung der Begriffe existiert. Um dieses Textstück nach dem Konzept von Wissensrepräsentationen auszudrücken, müssen die auftretenden Begriffe und ihre logische Beziehung durch ein maschinell interpretierbares Vokabular beschrieben werden [Antoniou, 2004]. Daher ist HTML für das Semantic Web nicht geeignet. Es wird eine Auszeichnungssprache benötigt, über die unterschiedliche Systeme miteinander über ein gemeinsames Vokabular kommunizieren können.

Aus dem Grund der fehlenden Maschineninterpretierbarkeit wurde bereits 1998 vom World Wide Web Consortium[12] (W3C) die Auszeichnungssprache XML[13] herausgegeben, die dazu dient andere Sprachen in einer standardisierten Weise zu repräsentieren, ohne eigene Befehle zur Verfügung zu stellen [Bray, 2006]. Der Inhalt eines HTML Dokuments kann, wie auch jede andere Auszeichnungssprache, durch XML Metainformation ausgedrückt werden. Jedes XML Dokument besitzt eine Wurzel und Elemente, die jeweils zwischen zwei Tags stehen. Diese Tags können verschachtelt sein und Werte besitzen. Das vorgestellte HTML Beispiel wird in Listing 2.2 als XML dargestellt.

Listing 2.2: XML Code Beispiel

```
<?xml version="1.0"?>
```

[12] http://www.w3.org/ (Stand: 01.10.08)
[13] http://www.w3.org/TR/2006/REC-xml-20060816/ (Stand: 15.09.08)

```
<Laender Asiens>
  <Land name="Japan">
    <Hauptstadt>Tokio</Hauptstadt>
  </Land>
</Laender Asiens>
```

Wie auch im HTML Ausschnitt wird Tokio als Hauptstadt von Japan und Japan als Land Asiens beschrieben. Hierzu wurden unterschiedliche Tags verwendet. Während Tokio von dem Tagpaar *<Hauptstadt></Hauptstadt>* eingeschlossen wird, wurde für Japan ein ebenso möglicher Tag *<Land></Land>* mit einem Attributwert *Japan* verwendet. Es ist ersichtlich, dass die Verschachtelung der Begriffe einer Baumstruktur entspricht in der Tokio unter Japan, und Japan unter Länder Asiens steht. XML stellt damit eine standardisierte Syntax zur Verfügung, enthält aber noch keine Hinweise, wie diese Information zu verarbeiten ist. Um den Codeausschnitt zu verwerten, muss erst geklärt werden, welche Tags in einem Dokument erlaubt sind und was sie bedeuten. Im Fall von HTML wird dies durch die Extensible HyperText Markup Language (XHTML)[14] spezifiziert. XHTML entspricht einer Neudefinition von HTML in XML und folgt somit den Syntaxregeln von XML, verwendet aber dieselben Tag Ausdrücke wie HTML [Klein, 2001]. In XHTML werden offene Tags wie *
* durch geschlossene Anweisungen ersetzt und andere problematische Eigenschaften von HTML, wie etwa die erlaubte Groß- bzw. Kleinschreibung, neu geregelt. Dies erleichtert es das Dokument maschinell zu verarbeiten.

Neben den HTML Tags sind aber auch andere beliebige Vokabulare möglich. Im gezeigten Beispiel wird ein Vokabular mit den Tagnamen *<Laender Asiens>, <Land>, <Hauptstadt>* verwendet. Um ein solches Vokabular festzulegen, bedient man sich einer Schemensprache, die für die bestehende Syntax die verwendbaren Begriffe und Attribute festlegt. Für XML existieren beispielsweise die Schemensprachen DTD[15] und XML Schema[16]. In DTD könnte definiert werden, dass ein *Land* einen Attributwert haben muss und ein Kind *Hauptstadt* besitzen kann. XML Schemen erlauben bei der Begriffsdefinition mehr Freiheiten und besitzen den Vorteil, selbst im XML Syntax definiert zu sein [Klein, 2001]. Obwohl XML und die zugehörigen Schemensprachen bereits einige für das Semantic Web wichtige Eigenschaften mitbringen, sagen sie immer noch nichts über die Bedeutung der Begriffe der XML Syntax aus. Um logische Relationen und Schlüsse aus den Begriffen ziehen zu können, ist es notwendig, eine weiterentwickelte Auszeichnungssprache auszuwerten.

[14] http://www.w3.org/TR/xhtml11/ (Stand: 15.09.08
[15] http://edition-w3c.de/TR/2000/REC-xml-20001006/ (Stand: 15.09.08)
[16] http://www.edition-w3c.de/TR/2001/REC-xmlschema-0-20010502/ (Stand: 15.09.08)

2.3.2 RDF und RDFS

1999 schlug die W3C Organisation den Resource Description Framework (RDF)[17] als Metasprache vor, um Information im Internet zu repräsentieren [Manola, 2004]. RDF wird dazu verwendet, semantische Beziehungen darzustellen. Dazu wird ein einfaches Datenmodell verwendet, indem Aussagen der Form *"Ein Ding S hat eine Eigenschaft P mit einem Wert O"* getroffen werden. Hierbei kann S als Subjekt, P als Prädikat und O als Objekt verstanden werden. Da eine RDF Aussage stets aus diesen drei Elementen besteht wird sie als *triple* bezeichnet. Anstelle von Subjekten und Prädikaten spricht man bei einer Aussage in RDF von Ressourcen und Literalen. Ressourcen können sowohl als Subjekt, Prädikat oder Objekt fungieren und stellen ein beliebiges im Internet auftretendes Element dar. Jede Ressource muss lediglich durch einen eindeutigen Identifikator (URI) gekennzeichnet sein. Literale sind konkrete Werte, die als Objekt O in einer RDF Aussage auftreten [Beck, 2004].

In Listing 2.3 werden drei RDF Aussagen definiert, die wiederum die Beziehung von *Tokio* zu *Japan*, und *Japan* zu *Länder Asiens* beschreiben. Die dritte Aussage verwendet als Objekt ein Literal, nämlich eine Telefonvorwahl mit dem Wert *0081*. Auf diese Weise können beliebige Beziehungen dargestellt werden. Derartige RDF Aussagen lassen sich durch einen gerichteten Graf darstellen. Darin wird jede Ressource durch ein Oval und jedes Literal durch eine Box dargestellt. Eigenschaften von Ressourcen werden durch Pfeile gekennzeichnet [Manola, 2004]. Abbildung 2.1 zeigt den gerichteten Graf für das genannte Beispiel. Um die Ressourcen und ihre Eigenschaften darzustellen wurden geeignete fiktive URIs verwendet, die auf beliebige Internet Inhalte verweisen können.

Listing 2.3: RDF Aussagen

```
1 Länder Asiens hat ein Land mit dem Wert "Japan"
2 Japan hat eine Hauptstadt mit dem Wert "Tokio"
3 Japan hat eine Vorwahlnummer mit dem Wert "0081"
```

RDF bietet an und für sich keinerlei Vorgaben an die Syntax von Aussagen und liefert lediglich ein Modell zur Beschreibung von Metadaten. Um die Syntax von RDF Aussagen festzulegen ist eine eigene RDF Syntax erforderlich. Hier bietet sich der XML Standard an, in dem wie jede beliebige Auszeichnungssprache auch RDF Aussagen formuliert werden können. Neben XML existieren auch andere RDF Syntaxen wie die von Tim Berners-Lee entwickelte und ebenso gebräuchliche Syntax N3[18]. N3 zielt besonders darauf ab RDF Aussagen einfacher und kürzer darstellen zu können, als es in XML

[17] http://www.w3.org/RDF/ (Stand: 15.09.08)
[18] http://www.w3.org/DesignIssues/Notation3.html (Stand: 29.06.08)

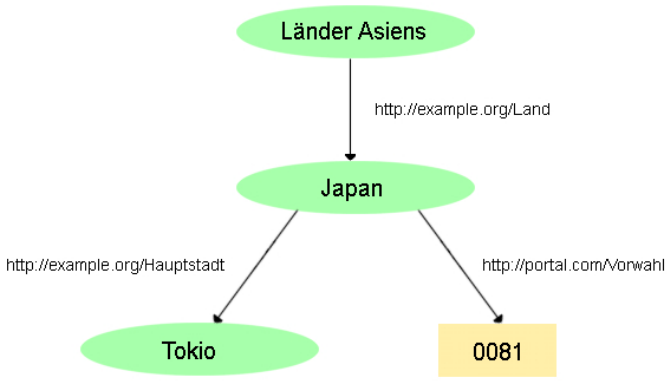

Abbildung 2.1: Ein gerichteter Graf für 3 RDF Aussagen.

möglich ist. Um die in RDF spezifizierten Aussagen verarbeiten zu können, ist zusätzlich ein Vokabular nötig, welches die verwendeten Begriffe definiert. Wie auch bei XML ist hierzu eine Schemensprache nötig. Für RDF wurde 2000 die RDF Vokabular Beschreibungssprache RDF Schema (RDFS)[19] vom W3C vorgeschlagen [Brickley, 2004]. Wenn ein RDF Schema neben dem Vokabular auch die nötigen Regeln zur Interpretation der vorkommenden Ressourcen enthält, wird es zu einer Ontologie. Neben RDFS gibt es noch zahlreiche anderen Schemensprachen die verwendet werden können. Darunter fällt auch die Web Ontology Language (OWL)[20], die als Erweiterung von RDFS angesehen werden kann und einige zusätzliche Möglichkeiten bei der Erstellung einer Ontologie bietet.

RDFS ist im Prinzip ein Datentyp System für RDF, das selbst in RDF geschrieben wird. Es definiert verschiedene Klassen, denen Domäne spezifische Eigenschaften zugeordnet werden können. Durch unterschiedliche Primitive kann eine hierarchische Struktur durch Klassen und Subklassen sowie Eigenschaften und Subeigenschaften aufgebaut werden. Daneben können möglichen Kombinationen von Klassen und Eigenschaften eingeschränkt und gewisse Ressourcen als Instanzen von Klassen spezifiziert werden. Die wichtigsten RDFS Modeling Primitive können folgendermaßen zusammengefasst werden [Brickley, 2004]:

Klassen: Klassen sind selbst Ressourcen und besitzen eine eindeutige URI. Man unterscheidet folgende Klassentypen:

[19] http://www.w3.org/TR/rdf-schema/ (Stand: 15.09.08)
[20] http://www.w3.org/TR/owl-features/ (Stand: 15.09.08)

- *rdfs:Resource* bezeichnet eine beliebige Ressource und damit praktisch jegliches im Internet mögliche Element.
- *rdfs:Class* sind Klassen von Ressourcen die Klassen sind. rdfs:Class ist eine Instanz von rdfs:Class.
- *rdfs:Literal* ist eine Klasse für Literale wie Strings oder Integer Werte.
- *rdfs:Property* stellt die Klasse für alle Eigenschaften dar.
- *rdfs:Statement* repräsentiert die Klasse aller RDF Aussagen, die durch triples gebildet werden.

Eigenschaften: Das Konzept einer RDF Eigenschaft entspricht dem Prädikat einer Relation zwischen dem Subjekt und Objekt einer RDF Aussage. Die wichtigsten RDFS Eigenschaften sind:

- *rdfs:type* deklariert eine Ressource als Instanz einer Klasse.
- *rdfs:subClassOf* verbindet eine Klasse mit einer ihr übergeordneten Klasse.
- *rdfs:subPropertyOf* verbindet eine Eigenschaft mit einer ihr übergeordneten Eigenschaft. Durch subClassOf und subPropertyOf wird es möglich, hierarchische Zusammenhänge zu beschreiben.
- *rdfs:domain* spezifiziert die Domäne einer Eigenschaft P, also alle Ressourcen, die als Subjekt einer RDF Aussage für ein Prädikat P auftreten können.
- *rdfs:range* spezifiziert alle Ressourcen, die als Wert einer RDF Aussage für eine Eigenschaft verwendet werden können.

Im Beispiel aus Abbildung 2.1 könnte somit ein RDF Schema mit der Ressource *Land* und den Eigenschaften *Hauptstadt* und *Vorwahl* definiert werden. Die Eigenschaften hätten die gemeinsame Domäne *Land* und als mögliche Werte Literale. Mit diesem Schema kann *Japan* als Instanz von *Land* mit den entsprechenden Werten definiert werden. Damit wäre nun eine Interpretation und dementsprechend eine Validierung der Daten möglich. Da RDFS jedoch die Definition jedes beliebigen RDF Schemas erlaubt, ist es wichtig, dass sich Internetbetreiber, die miteinander kommunizieren wollen, auf ein einheitliches Schema einigen.

2.3.3 Web Ontology Language

Anstelle von RDFS können auch andere Schemensprachen verwendet werden um das Vokabular und die logischen Beziehungen der Ressourcen zu repräsentieren. Da die Ausdruckskraft von RDFS auf eine hierarchische Struktur von Klassen und Eigenschaften sowie einigen Domäne und Werteeingrenzungen beschränkt ist, wurde die Notwendigkeit für stärkere Schemensprachen schon früh von der Web Ontology Working Group des W3C erkannt [Antoniou, 2004]. Aus diesem Wunsch heraus entstand die Web Ontology Language als Nachfolger von DAML+OIL[21], die nach mehrjähriger Entwicklung 2004 vom W3C herausgegeben wurde[22]. OWL erweitert die Möglichkeiten, um Eigenschaften und Klassen von RDF Aussagen zu beschreiben [Antoniou, 2004]. beschreibt die Limitationen von RDFS die durch OWL beseitigt werden können:

Lokale Werteeinschränkungen: In RDFS können Werte von RDF Aussagen durch die Klasse rdfs:range eingeschränkt werden. Es ist aber nicht möglich, diese Einschränkung nur für eine Teilmenge von Klassen zu definieren.

Zerlegung von Klassen: Oberklassen lassen sich nicht in Subklassen zerlegen, etwa um die Klasse *Mensch* in die Klassen *Mann* und *Frau* zu zerlegen. In RDFS ist es nur möglich, Unterklassen hierarchisch zu spezifizieren.

Boolesche Kombinationen: Klassen können in RDFS nicht durch boolesche Kombinationen wie *Vereinigung, Komplement* und *Durchschnitt* kombiniert werden.

Einschränkung der Kardinalität: Es ist nicht möglich in RDFS festzulegen, dass eine bestimmte Eigenschaft eine klare Anzahl an Werten besitzen muss. Ein Mensch etwa hat genau zwei Elternteile und eine entsprechende Eigenschaft müsste daher genau zwei Werte besitzen.

Spezielle Charakteristiken: Manchmal kann es hilfreich sein Eigenschaften untereinander transitiv (wie *größer als*), singulär (wie *wird verwendet von*) oder invers (wie *verwendet und wird verwendet von)* in Beziehung zu setzen.

Diese Einschränkungen werden durch OWL behoben. Um die Komplexität der Sprache je nach Anwendungsfall gering zu halten, wurden drei Untersprachen definiert, die die Freiheiten zum Teil einschränken [McGuiness, 2004]:

[21] http://www.daml.org/ (Stand: 15.09.08)
[22] http://www.w3.org/2001/sw/WebOnt/ (Stand: 29.06.08)

OWL Lite: OWL Lite zielt auf eine einfache Handhabung und hierarchische Zusammenhänge zwischen Ressourcen ab. Obwohl Kardinalitäten erlaubt werden, können diese nur die Werte 0 oder 1 besitzen. OWL Lite besitzt eine geringere formelle Komplexität als OWL DL und erlaubt eine einfache Erstellung von Taxonomien.

OWL DL: OWL DL richtet sich an BenutzerInnen, die mehr Ausdruckskraft mit einer einfachen Verarbeitbarkeit und Entscheidbarkeit verbinden wollen. Das bedeutet, dass alle Folgerungen in einer endlichen Zeit verarbeitbar sein müssen. OWL DL benutzt alle OWL Konstrukte jedoch unter gewissen Bedingungen. Beispielsweise kann eine Klasse nicht als Instanz einer anderen Klasse fungieren.

OWL Full: OWL Full bietet eine komplette syntaktische Freiheit um RDF Aussagen zu beschreiben und verzichtet somit auf die Garantie der maschinellen Verarbeitbarkeit.

Als Sprachkonstrukte bietet OWL wie auch RDFS, Klassen (rdfs:class), Eigenschaften (rdfs:property), Subklassen (rdfs:subClassOf), Subeigenschaften (rdfs:subPropertyOf), Domäne- (rdfs:domain) und Werteeinschränkungen (rdfs:range). Diese Grundfunktionalitäten werden durch einige zusätzliche Sprachkonstrukte erweitert. OWL Lite enthält bereits einige Features um die Gleichheit oder Ungleichheit von Ressourcen zu spezifizieren, zusätzliche Identifikatoren um die Charakteristiken von Eigenschaften zu beschreiben, zusätzliche Einschränkungsmöglichkeiten und Klassenüberschneidungen. OWL DL und OWL Full erweitern dies noch durch Boolsche Kombinationen und zahlreichere Werte und Kardinalitätsbedingungen. Eine komplette Auflistung der Sprachkonstrukte von OWL befindet sich unter [McGuiness, 2004].

Damit bietet die Web Ontology Language einen hohen Grad an Freiheit bei der Spezifikation von semantischen Beziehungen. Jedoch ist für eine sinnvolle Verarbeitung die Voraussetzung gegeben, dass unterschiedliche Applikationen gleiche Ontologien verwenden, da jede Neuspezifikation einer Ontologie bedeutet, dass Daten anderer Ontologien erst angepasst werden müssen. Aus diesem Grund haben sich mit dem Aufkommen des Semantic Web diverse Schemengemeinschaften gebildet die daran arbeiten, einheitliche Ontologien für unterschiedlichste Anwendungsfälle zu entwickeln. Auf der anderen Seite entstehen vermehrt Werkzeuge um Ontologien miteinander zu verknüpfen, siehe [Noy, 2000], [Noy, 1999] oder [Kim, 2005]. Einige Autoren sehen in der verteilten Entwicklung von semantischen Beschreibungen eine Schwäche des Semantic Web, die letztlich zum Fall des Konzeptes führen könnte [Diestelkamp, 2005]. Jedoch deutet die wachsende Zahl an Anwendungen darauf hin, dass sich das Semantic Web vermehrt zu etablieren beginnt.

2.3.4 Semantische Abfragesprachen

Um nun spezifische Informationen aus den semantischen Daten zu gewinnen, müssen diese auf irgendeine Art und Weise gefiltert werden. Dies ist durch den Einsatz von RDF Abfragesprachen möglich, die dazu dienen, die Gesamtmenge der vorliegenden Daten auf Basis einer Abfrage (Query) einzugrenzen. Obwohl es viele Abfragesprachen für RDF gibt, hat sich SPARQL[23] aufgrund der W3C Empfehlung de facto als Standard etabliert [Prud'hommeaux, 2008]. Die Formulierung eines SPARQL Queries erinnert an die Struktur der Datenbank Abfragesprache SQL[24]. Der Aufbau einer solchen Abfrage wird in Listing 2.4 gezeigt. Zunächst können zur besseren Lesbarkeit Präfixe definiert werden um die Angabe von URIs abzukürzen. Danach werden mit der Anweisung *SELECT* die gewünschten Ausgaben definiert, die im *WHERE* Teil näher spezifiziert werden. In diesem Beispiel werden die Hauptstädte und Länder aller RDF Triples selektiert, die semantisch dem Kontinent Asien zugeordnet wurden. Hierzu wurden die Variablen x und y, die entsprechende Inhalte abfragen, miteinander in Beziehung gesetzt. Die Formulierung entspricht der Abfrage einer Datenbank und erlaubt einen hohen Freiheitsgrad.

Listing 2.4: SPARQL Query Beispiel

```
1 PREFIX uri: <http://example.com/exampleOntology>
2 SELECT ?capital ?country
3 WHERE {
4   ?x uri:cityname ?capital.
5   ?y uri:countryname ?country.
6   ?x uri:isCapitalOf ?y.
7   ?y uri:isInContinent uri:asia.
8 }
```

2.4 Anwendungen des Semantic Web

Mit der fortlaufenden Entwicklung der Technologien RDF, RDFS und OWL wurde in den letzten Jahren ein Grundgerüst geschaffen, um Tim Berners-Lees Vision eines semantischen Web zu ermöglichen. Obgleich die Anzahl der bestehenden Anwendungen und aktiven Arbeitsgruppen noch überschaubar ist, wächst deren Zahl fortlaufend. Auch wurden in den letzten Jahren immer mehr Werkzeuge und Tools zur Erstellung semantischer

[23] http://www.w3.org/TR/rdf-sparql-query/ (Stand: 29.06.08
[24] http://www.w3schools.com/sql/sql_intro.asp (Stand: 29.06.08)

Inhalte entwickelt. Neben semantischen Entwicklungsumgebungen existieren RDF Generatoren, OWL Validatoren, Semantische Suchmaschinen oder Tools zur Verknüpfung von Ontologien [25]. In diesem Abschnitt werden nun einige Semantic Web Anwendungen vorgestellt um eine Idee des aktuellen Stands der Möglichkeiten zu vermitteln. Danach folgt eine kurze Liste zu interessanten Semantic Web Anwendungen, deren genaue Darstellung jedoch den Rahmen dieser Arbeit sprengen würde.

Freebase

Freebase[26] von Metaweb Technologies [27] ist eine große offene Datenbank über das verteilte Wissen im Internet. Im Gegensatz zu Internet Enzyklopädien wie Wikipedia setzt die Software jedoch eine Datenbank ein, in der Daten semantisch miteinander verknüpft sind. Bei der Erstellung und Verwaltung ihrer Inhalte folgt die Plattform den Web 2.0 Prinzipien. Information wird in Form von kurzen Aussagen zu einem neuen oder bestehenden Thema hinzugefügt. Für ein neues Thema können über ein Interface auch eigene Schemen definiert werden. Freebase enthält einige Applikationen, um das vorhandene Wissen zu verwerten. Hierzu wird eine eigene Abfragesprache für Metadaten (Meta Query Language) zur Verfügung gestellt. Es wird also zentral vorliegendes Wissen genutzt um das Semantic Web in einem geschlossenen Rahmen umzusetzen. Das Projekt versucht die Idee von Internet Enzyklopädien in einem semantischen Zusammenhang umzusetzen, um neue Anwendungen zu ermöglichen, die Informationen aus verteilt vorliegenden Daten extrahieren können. Abbildung 2.2 zeigt einen Screenshot des Portals von Freebase.

DBpedia

DBpedia[28] (Abbildung 2.3) ist eine Datenbank die versucht, das Wissen im Internet semantisch zu annotieren und frei verfügbar zu machen. Zu diesem Zweck werden sämtliche Begriffe aus Wikipedia mit einer einheitlichen Ontologie beschrieben und über verschiedene RDF Syntaxen verfügbar gemacht. Im Gegensatz zu Freebase, unterstützt das Projekt keine eigene Community, um das enthaltene Wissen zu erweitern. Stattdessen wird versucht, Inhalte über zahlreiche Schnittstellen zugänglich zu machen. Damit

[25] http://esw.w3.org/topic/SemanticWebTools (Stand: 29.06.08)
[26] http://www.freebase.com (Stand: 29.06.08)
[27] http://metaweb.com (Stand: 29.06.08)
[28] http://www.DBpedia.org (Stand: 29.06.08)

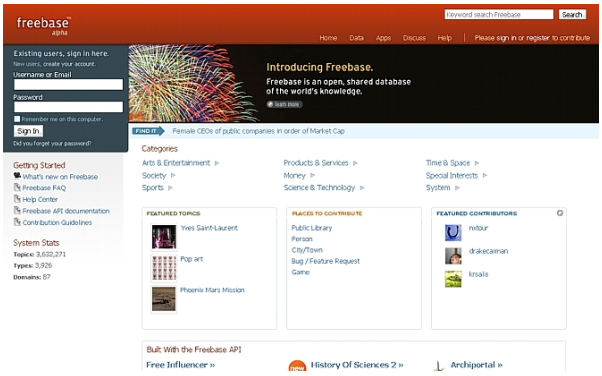

Abbildung 2.2: Freebase (http://www.freebase.com)

bietet DBpedia ein gutes Fundament, um erste Anwendungen für das Semantic Web zu ermöglichen ohne dem Problem gegenüber zu stehen, Ontologien unterschiedlicher Quellen verknüpfen zu müssen.

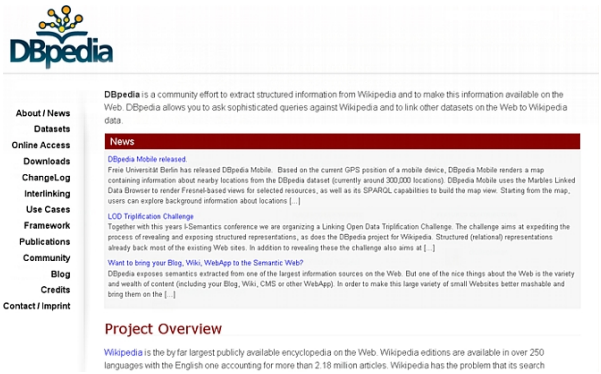

Abbildung 2.3: DBpedia (http://www.dbpedia.org)

Powerset

Powerset[29] (Abbildung 2.4) stellt einen semantischen Suchdienst zur Verfügung, der in der aktuell vorliegenden Version noch auf die Inhalte von Wikipedia und Freebase beschränkt ist. Fragen können dabei auf natürliche Weise formuliert werden. Um Daten verarbeiten zu können, werden Dokumente semantisch beschrieben, nachdem sie durch

[29] http://www.powerset.com (Stand: 29.06.08)

eine eigens am Palo Alto Research Center entwickelte Analysetechnik für natürliche Sprachen verarbeitet wurden. Beispielsweise können so Abfragen der Form *Wie heißt die Hauptstadt von Österreich?* oder *Wann wurde Wien gegründet?* getroffen werden. Anders als bei gewöhnlichen Suchmaschinen liefert die Applikation nicht nur eine Liste mit Treffern. Stattdessen wird die gesuchte Information aus den vorliegenden Daten extrahiert. In Anlehnung an herkömmliche Internetsuche wird zusätzlich eine Linkliste mit passenden Artikeln angeboten. Die Suchmaschine zeigt wie verteilt vorliegende Daten genutzt werden können, um daraus spezifische Informationen zu extrahieren, wenngleich dies noch unter Verwendung weniger Quellen geschieht.

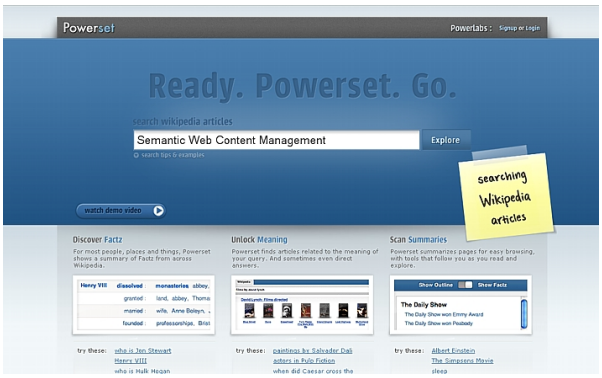

Abbildung 2.4: PowerSet (http://www.powerset.com)

Adaptive Blue

Adaptive Blue[30] (Abbildung 2.5) ist eine Firefox[31] Erweiterung um Bookmarks eines Browsers nach semantischen Gesichtspunkten anzuordnen. Bookmarks, oder Lesezeichen, sind lokale Verweise auf Internetseiten, die von BenutzerInnen im Browser gespeichert werden können. Diese können dann in selbst erstellten Ordnern kategorisiert werden. Mit dem Organizer von AdaptiveBlue lassen sich Verbindungen zwischen Kategorien und Websites entdecken und Bookmarks automatisch anordnen. Um das zu ermöglichen, werden semantische Beschreibungen der Webseiten Betreiber verwendet und ausgewertet. Die gespeicherten Bookmarks können außerdem mit anderen BenutzerInnen geteilt werden, um gemeinsame Interessen zu entdecken.

[30] http://www.adaptiveblue.com (Stand: 29.06.08)
[31] http://www.mozilla-europe.org/de/firefox/ (Stand: 29.06.08)

Abbildung 2.5: AdaptiveBlue (http://www.adaptiveblue.com)

Tabelle 2.1 enthält eine Auswahl kommerzieller, oder gemeinnütziger Semantic Web Anwendungen, die bereits heute verfügbar sind. Die Zahl der vorhandenen Anwendungen zeigt, dass die Konzepte und Technologien durchaus schon Verbreitung im Internet finden. Obwohl die meisten Angebote auf geschlossene Plattformen beschränkt sind oder noch in Entwicklung stehen ist es offen, welche Bedeutung das Semantic Web in Zukunft gewinnen wird. Die Voraussetzungen sind in jedem Fall bereits gegeben.

Freebase	Beschreibung	Semantische Wissensdatenbank
	Typ	Creative Commons Lizenz[32] (BY)
	Technologien	Meta Query Language (JSON Basis)
Powerset	Beschreibung	Suchmaschine für natürliche Sprache
	Typ	Kommerziell / Freie Nutzung
	Technologien	Nicht einsehbar
Twine	Beschreibung	Semantisches Tagging
	Typ	Kommerziell / Freie Nutzung
	Technologien	Nicht einsehbar
AdaptiveBlue	Beschreibung	Plugin für SmartLinks
	Typ	Kommerziell / Freie Nutzung
	Technologien	Firefox Plugin
DBpedia	Beschreibung	Semantische Wissensdatenbank
	Typ	Opendefiniton Lizenz[33] (Open Data)
	Technologien	PHP5, SQL, SPARQL
Hakia	Beschreibung	Semantische Suchmaschine
	Typ	Kommerziell / Freie Nutzung
	Technologien	Nicht einsehbar
Swoogle	Beschreibung	Semantic Web Suchmaschine
	Typ	Creative Commons Lizenz[34]
	Technologien	Java, Apache, Tomcat, PHP, mySQL
True Knowledge	Beschreibung	Semantische Wissensdatenbank
	Typ	Kommerziell / Freie Nutzung
	Technologien	True Knowledge API
Tripit	Beschreibung	Semantisch unterstützte Reiseplanung
	Typ	Kommerziell / Freie Nutzung
	Technologien	Nicht einsehbar
Spock	Beschreibung	Semantisch unterstützte Personensuche
	Typ	Kommerziell / Freie Nutzung
	Technologien	Spock API, XML
Esolda	Beschreibung	Semantische Produktsuche
	Typ	Kommerziell / Freie Nutzung
	Technologien	Nicht Einsehbar

Tabelle 2.1: Auswahl von Semantic Web Anwendungen

3 Semantic Web und Content Management

Content Management Systeme (CMS) sind heute im Internet weit verbreitet, um die immer größer werdende Zahl an unterschiedlichen Inhalten zu organisieren. Durch eine Trennung von Inhalt und Layout werden RedakteurInnen entlastet, da sich ein System um die technische Umsetzung der Struktur einer Website kümmert. Daten werden über Metainformationen aus einer Datenbank dynamisch zusammengestellt und vermeiden so umständliche Updateprozesse, da Änderungen an Seitenelementen im gesamten System übernommen werden. Während in den letzten Jahren viele der Kernkonzepte des Web 2.0 in Web Content Management Systemen Einzug fanden, sind semantische Dienste in diesem Zusammenhang noch weitgehend unbekannt. Dennoch besitzen Systeme meist das technische Grundgerüst, um semantische Dienste umsetzen zu können. Schon heute wird in einigen Arbeitsgruppen, wie der Drupal Semantic Web Group[1], an Erweiterungen zum Import oder Export von RDF Dateien gearbeitet, um etwa dezentral gespeicherte Information in ein System integrieren zu können.

In diesem Kapitel wird die allgemeine Funktionsweise von Content Management Systemen vorgestellt und das Potential von semantischen Technologien anhand von Anwendungsbeispielen erläutert. Danach folgt eine Analyse, ausgewählter Content Management Systeme, um die Verfügbarkeit dieser Technologie zum jetzigen Stand auszuwerten. Dabei werden die Systeme auf ihr technisches Grundgerüst, die verfügbaren semantischen Ansätze und aktuelle Arbeitsgruppen zur Entwicklung neuer Erweiterungen untersucht. Das Kapitel endet mit einer Diskussion der Ergebnisse und einem Ausblick auf zukünftige Entwicklungen im Bereich des Semantic Web Content Management.

[1] http://groups.drupal.org/semantic-web (Stand: 15.09.08)

3.1 Grundlagen

Der Begriff des Content Management ist ein erst sehr junger Begriff der sich mit der Entwicklung des Internet stetig weitergebildet hat, da diese auch eine stetige Neudefinition von Content und dessen Management erforderlich macht[Gilbane, 2000]. Unter Content versteht man heute im Zusammenhang mit Webauftritten nicht mehr nur einfache multimediale Inhalte wie Text, Bilder und audiovisuelle Daten. Durch die technologische Entwicklung in den letzten Jahren wurden die Anforderungen an die Verwaltung von Inhalten deutlich gesteigert. Während das Internet in seiner ursprünglichen Form aus hauptsächlich statischen Dokumenten bestand, sind heute dynamische Programmcodes, zahlreiche E-Commerce Anwendungen und dazugehörende Datenbankverknüpfungen entstanden. Neben Multimedia Streaming, audiovisuellen Daten und Hypermedia, ist die Bandbreite an Inhalten enorm. Auch die Zahl der verfügbaren Dokumente ist in den letzten Jahren explosionsartig gestiegen und macht eine systematische Organisation dieser Inhalte immer notwendiger. Aufgrund der Vielfalt an unterschiedlichen Inhalten ist es heute sinnvoller, als Content die Summe aller veröffentlichten digitalen Assets einer Website zu bezeichnen. Das Management dieser Assets spiegelt sich in der Bereitstellung von Definition und Kontrolle eines Workflows wider, der den Ablauf einzelner Arbeitsschritte wie etwa bei redaktioneller Arbeit regelt [Zschau, 2001].

Das Einsatzgebiet von Content Management Systemen reicht von der Strukturierung einfacher persönlicher Blogs, über redaktionelle Arbeit, bis hin zu großen Projektplattformen mit integrierten Communities und E-Commerce Lösungen. All diese Systeme haben gemein, dass der Kosten und Zeitfaktor des Publizierens von Information minimiert werden soll. Durch die Trennung von Inhalt und Programmcode, können RedakteurInnen Artikel einfach und schneller publizieren, da sich das System um die technische Realisierung dieses Prozesses kümmert. Diese Trennung bedeutet auch eine Vereinfachung der Aktualisierung von Inhalten, da Adaptionen des Layouts oder Umstrukturierungen in alle bestehenden Seiten vom System integriert werden. Eine Einteilung der BenutzerInnen in Benutzergruppen mit unterschiedlichen Rechten und Kompetenzen unterstützt zudem die Qualitätssicherung [Zschau, 2001].

Ein Versuch, die Aufgaben eines Content Management Systems zu erfassen, gibt die folgende Darstellung wieder [Kristöfl, 2003]. Hiernach gliedern sich die Aufgabenbereiche eines solchen Systems in drei Teile: Die Differenzierung von Content und Layout, die Bereitstellung eines Komponenten Management und die Integration eines Workflow Management. Diese Bereiche werden nun im Einzelnen vorgestellt:

Differenzierung von Content und Layout Die Trennung von Inhalt und technischem Hintergrund war einer der Kernaspekte, die zur Entwicklung von Content Management Systemen führten. Das Design und der Aufbau einzelner Websites werden meist in Templates (Vorlagen) definiert, die für beliebige Projektseiten verwendet werden können. Nach der Erstellung des Inhaltes wird aus einem Template eine Seite dynamisch generiert und in der Menüstruktur des Systems integriert.

Bereitstellung eines Komponenten Management Die einzelnen Inhalte, wie Texte oder Bilder, werden mit Metadaten versehen und mit entsprechenden Verweisen in einer Komponenten Datenbank abgelegt. Auf diese Weise sind Texte, Bilder oder andere Inhalte an beliebigen Stellen wiederverwendbar und dynamische Verweise möglich.

Integration eines Workflow Management Ein individuell adaptierbares Workflow Management erleichtert die redaktionelle Arbeit an einer Projektseite. Artikel werden für einen bestimmten Zeitraum publiziert und automatisch archiviert. Ein Rechtesystem ermöglicht den individuellen Zugriff auf Programmteile und die Weiterleitung und vereinfachte Freischaltung von Artikeln.

Weitere Eigenschaften von Web Content Management Systemen betreffen Sicherheitsaspekte, die Erweiterbarkeit mittels Skriptsprachen und Modulen, Personalisierung von Webinhalten, XML-Fähigkeiten, Austausch von Inhalten zwischen Websites und Reportingfunktionen. Die Sicherheit im System ist wichtig, da bei Zugriffen von verschiedenen Personen Dokumente nicht gleichzeitig bearbeitet werden dürfen. Andernfalls können eingegebene Modifikationen verloren gehen. Durch die Personalisierung kann der Seitenaufbau an individuelle Wünsche angepasst werden. Dies ist durch die dynamische Generierung des Contents aus einer Datenbank gegeben. Die Integration von XML Import und Export ermöglicht einen Austausch von Daten zwischen unterschiedlichen Plattformen. Dies ist die Grundlage, um semantisch interpretierbare Daten zwischen Systemen auszutauschen [Zschau, 1999]. Abbildung 3.1 zeigt das Schema eines redaktionellen Content Management Systems.

3.2 Vom statischen Web zum Web Content Management

Die Herkunft der ersten WCMS lässt sich nicht klar zurückverfolgen[Gilbane, 2000]. Bereits Mitte der neunziger Jahre entstand eine Vielzahl an individuellen Projektlösungen,

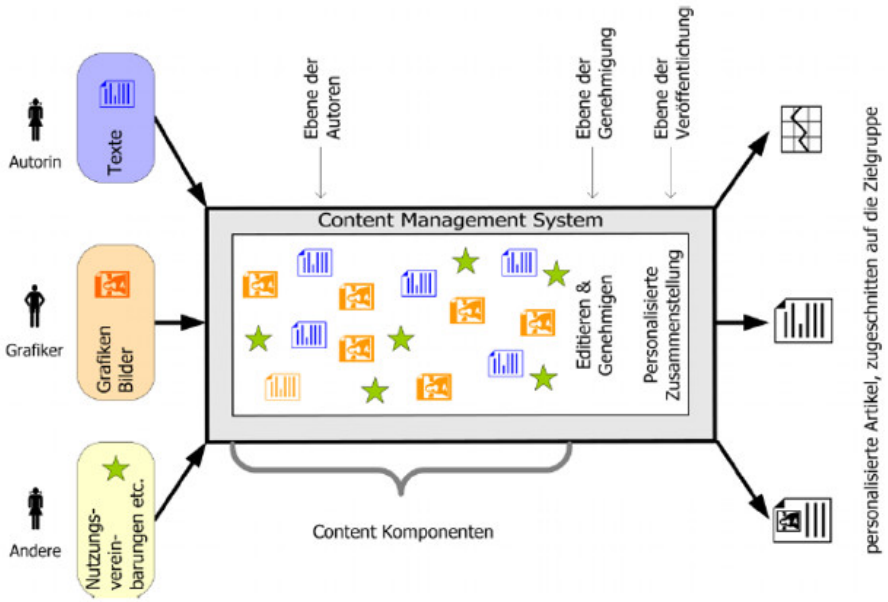

Abbildung 3.1: Das Schema eines Web Content Management Systems [Kristöfl, 2003].

um die Datenflut größerer Internetauftritte verwalten zu können. Die Entwicklung von Formatierungssprachen wie Cascade Stylesheet (CSS)[2] erlaubte erstmals eine Trennung von Inhalt und Layout einer Seite. Zudem wurden die ehemals statischen Seiten zunehmend durch dynamische serverseitige Scriptsprachen ersetzt. Es wurden Formalismen für die Erstellung und Verwaltung von Inhalten entwickelt, um redaktionelle Arbeit zu erleichtern. Schließlich kam es auch zu mehr Automatisierung und aus ehemals einfachen Werkzeugen entstanden Systeme mit zahlreichen Funktionalitäten wie Rechtesystemen, die Personalisierung von Content, bis hin zur Entwicklung von weitreichenden Schnittstellen, um Content Management Systeme modular zu erweitern, oder einfache Informationen untereinander zu teilen [Zschau, 2001].

Die Verwendung des Begriffs Web Content Management System (WCMS) geht auf eine Pressemitteilung der amerikanischen Firma Vignette[3] zurück, die im Juli 1996 bereits die wesentlichen Kernelemente eines CMS definierte. Noch im selben Jahr griffen einige Unternehmen die vorgestellten Konzepte auf und veröffentlichten einige der frühen Content Management Systeme [Doyle, 2004]. Das heute zu den populärsten Open Source

[2] http://www.w3.org/Style/CSS/ (Stand: 01.08.08)
[3] http://www.vignette.com/ (Stand: 01.08.l0)

Systemen gehörende CMS Typo3[4], wurde von Kasper Skârhoj 1997 aus dem primären Grund ins Leben gerufen, Content und Design voneinander abzugrenzen und wurde 2000 als vollwertiges System in der Beta Version veröffentlicht [Typo3, 2006]. Im selben Jahr entwickelten sich die beiden heute zu den bekanntesten CMS zählenden Systeme Drupal[5] und Mambo[6] (aus dem sich das System Joomla![7] abspaltete) aus komplett unterschiedlichen Online Community Projekten. Mambo wurde im Jahr 2000 noch kommerziell entwickelt und gründete sich als Mambo Foundation 2001 unter der General Public License (GPL)[8] neu[MAMBO, 2006]. Drupal entstand als eine kleine Community Plattform für StudentInnen und wurde als Privatprojekt an der Universität Antwerpen von Drias Buytaert und Hans Snijder entwickelt, um studentische Aktivitäten zu planen. Seit 2001 ist die Software offen verfügbar und hat sich zu einem WCMS entwickelt, das sich vor allem durch seinen sozialen Charakter von anderen Projekten abhebt [Drupal, 2006].

Content Management ist heute ein wichtiger Bestandteil des Internet geworden, was sich aus der hohen Zahl an Unternehmen ablesen lässt, die Systeme zur Verwaltung von Webseiten entwickeln. Alleine zu Beginn des Jahres 2008 zählte die CMS Vergleichsplattform CMS Matrix[9] über 880 unterschiedliche Angebote für die Verwaltung einfacher Blogs, über dynamische Asynchronous Java and XML (AJAX)[10] betriebene Websites, bis hin zu großen kommerziellen Plattformen für Unternehmen. Die überwiegende Mehrheit der Systeme basiert auf der Kombination der Scriptsprache PHP[11] und der Datenbank MySQL[12]. Daneben existieren aber auch Systeme auf anderen Sprachen wie Python[13] oder Java[14] und ermöglichen so eine breite Auswahl nach individuellen Bedürfnissen.

3.3 Web Content Management und Web 2.0

Klassische Content Management Systeme bilden das redaktionelle System einer Zeitung ab. Darin werden Artikel verfasst, publiziert und archiviert. Durch die zunehmende Be-

[4] http://www.typo3.com (Stand: 01.08.08)
[5] http://www.drupal.org (Stand: 01.08.08)
[6] http://www.mambo-foundation.org (Stand: 01.08.08)
[7] http://www.joomla.org (Stand: 01.08.08
[8] http://www.gnu.org/licenses/gpl-3.0.html (Stand: 15.09.08)
[9] http://www.cmsmatrix.org (Stand: 01.08.08)
[10] http://www.adaptivepath.com/ideas/essays/archives/000385.php (Stand: 15.09.08)
[11] http://www.php.net (Stand: 01.08.08)
[12] http://mysql.com (Stand: 01.08.08)
[13] http://python.org (Stand: 01.08.08)
[14] http://java.sun.com (Stand: 01.08.08)

deutung von sozialen Verknüpfungen des Webs und das Aufkommen des Web 2.0 muss dieses Bild jedoch überdacht werden. Communities und soziale Wissensplattformen (Wikis) gewannen deutlich an Bedeutung [Kristöfl, 2003]. Nicht nur konventionelle Desktop Programme wurden vermehrt ins Internet verlagert (wie etwa bei der Fotoverwaltungsplattform Flickr[15] oder Youtube[16]), auch die Art der Nutzung von Webseiten änderte sich. Communities beginnen sich zu sozialen Netzwerken zu entwickeln, in denen BenutzerInnen Inhalte selbst erzeugen und organisieren. Durch Tagging werden Mediainhalte mit Schlagworten versehen und so dynamische Organisationsformen geschaffen. Schon heute folgen viele Systeme diesem Paradigma und organisieren Information eher nach gemeinschaftlichen Kriterien als durch einen hierarchischen Publikationsprozess. Daraus ergibt sich ein interessanter Kontrast zwischen klassischem Content Management und Ansätzen die den neuen, als Web 2.0 bekannten Erscheinungen, im Internet folgen.

3.4 Semantic Web und Content Management

Die Integration von semantischen Diensten in WCMS steht hingegen noch am Anfang und ist weitaus weniger verbreitet. Dabei bringt ein Content Management System in den Grundzügen jene Eigenschaften mit, die für das Semantic Web essentiell sind. Die Grundlage für maschineninterpretierbare Information ist die strukturelle Organisation von Information [Koller, 2005]. Content Management bringt eine klare Organisationsform in die Fluten von angebotener Information, um mehr Kontrolle über diese Inhalte zu ermöglichen.

Eine semantische Beschreibung dieser Struktur stellt nun den nächsten Schritt dar. Dadurch eröffnen sich zahlreiche neue Anwendungsgebiete. Viele Arbeiten verweisen auf das hohe Potential, das sich aus der Integration semantischer Dienste in WCMS eröffnen würde [Beck, 2004], [Koller, 2005], [Pellegrini, 2006]. Der Austausch von Inhalten zwischen Systemen wäre nach sinnvollen Kriterien möglich und würde beispielsweise erlauben, Produktbeschreibungen eines Herstellers in E-Commerce Anwendungen zu integrieren, oder Angaben zu einem Artikel öffentlich verfügbar zu machen. Im semantischen Web könnten solche Informationen über intelligente Suchmaschinen gesucht und verarbeitet werden. Bereits heute gibt es semantische Suchmechanismen zur individuellen Zusammenstellung von Information auf Basis gewählter Eigenschaften. Die Suchmaschi-

[15] http://www.flickr.com (Stand: 01.08.08)
[16] http://youtube.com (Stand: 01.08.08)

ne Esolda[17] verwendet aufbereitete Produktdaten, um die Suche nach einem Artikel nach individuellen Kriterien einzugrenzen. Auf ähnliche Weise könnten sich ganze Websites auf Basis persönlicher Wünsche adaptieren lassen, um direkt auf BenutzerInnen zugeschnitten zu sein.

In [Pellegrini, 2006] werden die möglichen zukünftigen Anwendungen von semantischen CMS zusammengefasst:

Aus BetreiberInnensicht:

- Plattformunabhängige Contentsyndication zwischen verschiedenen Systemen.
- Zugriff auf dezentral gespeicherte Daten.
- Beschleunigung redaktioneller Prozessschritte durch (semi-)automatisierte Integration.
- Erleichterte Verwaltung der Inhalte durch Metamodelle.
- Kompilierbarkeit von Content.

Aus BenutzerInnensicht:

- Bessere Auffindbarkeit von Dokumenten und Reduktion von Suchzeiten.
- Erschließung von Themen aus unterschiedlichen Benutzerperspektiven und Interessen.
- Beschreibung komplexer Zusammenhänge mittels Kontextinformation.
- Suchprozessoptimierung durch hochgradige Vernetzungsmöglichkeiten.
- Barrierefreier Zugang zu Information durch standardisierte Metamodelle (Accessibility).

Ein Beispiel für die Anwendbarkeit semantisch verfügbarer Daten im Internet stellt die Firefox Erweiterung Piggy Bank[18] dar, mit der einzelne Bereiche von Websites über deren RDF Beschreibung individuell zusammengestellt werden können. Es werden Informationen aus unterschiedlichen Quellen zentral zusammengefasst und nach einem persönlichen Organisationsschema strukturiert. Mit der nötigen Bereitstellung von Inhalten in Form von RDF Daten könnten auch Web Content Management Systeme zukünftig hierzu in der Lage sein [Pellegrini, 2006].

Viele Systeme verwenden klar definierte Felder oder Formulare zur Beschreibung von Attributen, beispielsweise AutorIn und Thema eines Artikels. Diese Felder können durchaus als Teil eines Schemas betrachtet werden, das die Beschreibung des Inhaltes über Metadaten erlaubt [Koller, 2005]. Insofern stellt die Umwandlung von Inhalten in ein

[17] http://www.esolda.at/ (Stand: 01.08.08)
[18] http://simile.mit.edu/wiki/Piggy_Bank (Stand: 01.08.08)

RDF Schema einen Schritt dar, der durch einfache Scripts oder Erweiterungen denkbar ist. Natürlich muss das System die dafür nötigen Minimalanforderungen erfüllen. Um ein RDF Schema zu erzeugen, müssen Inhalte als valides XML Dokument darstellbar und auch an eine semantische Beschreibungssprache anpassbar sein. Für praktische Anwendungen ist zudem ein Import oder Export dieser Information nötig, um sie sowohl verfügbar zu machen, als auch in das System zu integrieren. Roggenbuck und Sperber [Roggenbuck, 2006] weisen außerdem darauf hin, dass für eine semantische Darstellung die Verfügbarkeit von stabilen Adressen eines Dokuments wichtig ist, damit definierte Metadaten auch eine langfristige Gültigkeit besitzen. Obwohl die meisten Content Management Systeme ihre Inhalte dynamisch erzeugen, verfügen viele Systeme über die Fähigkeit stabile URLs zu erzeugen.

Eine Technologie die bereits seit einigen Jahren auf dieser Basis existiert ist RDF Site Summary (RSS)[19]. RSS ist ein weitläufig eingesetztes RDF Schema, um Nachrichtenmeldungen zu beschreiben. Die Bereitstellung von Kurznachrichten über RSS ist weithin auch als RSS Feed bekannt. Dabei enthält ein Element eine Überschrift, einen kurzen Textabriss und einen Link zu der publizierenden Seite. Diese Nachrichten können über den Browser abonniert und gelesen, aber auch geparst und weiterverwendet werden. Anhand von RSS wird besonders gut sichtbar, wie ein RDF Schema über ein Content Management System erzeugt werden kann (siehe Abbildung 3.2). Die im System enthaltenen Formularfelder werden hierbei in ein auf XML aufsetzendes RDF Dokument umgesetzt und für andere Maschinen gespeichert. Verallgemeinert könnten beliebige Informationen auf diese Weise exportiert und an anderer Stelle wieder importiert werden.

3.5 Semantic Web Standards

Mittlerweile gibt es bereits einige Ansätze, um Content Management Systeme über einheitliche Konventionen an das Semantic Web heranzuführen. Während einige Systeme bereits über individuelle Schnittstellen verfügen, um Daten zwischen unterschiedlichen Plattformen des selben Systems auszutauschen, versuchen dritte Parteien, allgemeine Standards zu entwickeln, über die auch unterschiedliche Systeme miteinander kommunizieren und Daten austauschen können. In diesem Abschnitt werden mit SIOC[20] und Triplify[21] zwei derartige Initiativen vorgestellt, die möglicherweise einen wichtigen Schritt

[19] http://www.rssboard.org/rss-specification (Stand: 01.08.08)
[20] http://sioc-project.org (Stand: 29.06.08)
[21] http://triplify.org (Stand: 29.06.08)

Abbildung 3.2: Umsetzung von Labels in ein RSS Schema [Koller, 2005].

zur Vernetzung von Semantic Web Content Management Systemen beitragen werden.

3.5.1 SIOC

Die SIOC Gruppe besteht seit 2004 und ist seit 2007 ein W3C Mitglied und damit auf dem Weg einen einheitlichen Standard für die Verbindung von Blogging Systemen und Internet Foren zu schaffen. Zu diesem Zweck entwickelt die SIOC Gruppe eine Ontologie, die speziell auf die Struktur von Internetforen ausgerichtet ist. Damit die Schnittstelle funktioniert, muss für jedes Forensystem aber ein entsprechendes Plugin entwickelt werden, da sich verschiedene Foren selbstverständlich technisch unterscheiden. Solche Schnittstellen existieren inzwischen für einige in ihrer Zahl jedoch noch überschaubare CMS. Darunter fallen WordPress[22] und Drupal. Programmierschnittstellen (APIs) gibt es für die Programmiersprachen PHP und Java womit auch eigene Module und Anwendungen mit SIOC entwickelt werden können. Durch die Aufnahme in das W3C ist die SIOC Gruppe sicherlich auf einem guten Pfad, sich als Standard für den Datenaustausch in diesem Bereich zu festigen.

[22] http://www.wordpress.com (Stand: 01.08.08)

3.5.2 Triplify

Triplify ist ein kleines Plugin für CMS, um den in Datenbanken vorliegenden Inhalt als RDF verfügbar zu machen. Die Verbreitung der Idee geht jedoch noch eher langsam voran. Obwohl einige Projekte für die Integration bestehen, existieren zum Zeitpunkt dieser Arbeit lediglich Plugins für einige größere Content Management Systeme wie Typo3, phpBB[23], XOOPS[24], e107[25] oder Plone[26]. Triplify stellt sicherlich einen guten Ansatz dar, Systeme über ein einheitliches Vokabular miteinander zu verbinden und die wachsende Verbreitung könnte es zu einem Standard der Zukunft in diesem Bereich machen. Es bleibt aber abzuwarten, ob die Triplify Ontologie von der Großzahl der CMS EntwicklerInnen angenommen wird.

3.5.3 Individuelle Ontologien

Daneben wird vielerorts auch an individuellen Lösungen gearbeitet, um Systeme desselben Typs miteinander zu verknüpfen. Ein Beispiel hierfür stellt das Drupal RDF Schema[27] dar, das speziell auf die Struktur des Content Management Systems zugeschnitten ist. Dies eröffnet natürlich den Vorteil, dass die Inhalte des Systems besser verarbeitbar werden und Module entwickelt werden können, um Daten zumindest semiautomatisch auszutauschen. Dafür bedeuten individuelle Ontologien eine weitere Hürde bei der Vernetzung unterschiedlicher Systeme, und verlieren in diesem Punkt gegen die vorgestellten Drittanbieter. Dennoch zeigt die Vielfalt an Arbeitsgruppen und Richtungen, dass durchaus Bestrebungen existieren, um das Semantische Web im Bereich des Content Management zu etablieren und einige dieser Projekte auf dem Weg sind, zu anerkannten Standards zu werden.

[23] http://www.phpbb.com/ (Stand: 01.10.08)
[24] http://www.xoops.org/ (Stand: 01.10.08)
[25] http://e107.org/news.php (Stand: 01.10.08)
[26] http://plone.org/ (Stand: 01.10.08)
[27] http://groups.drupal.org/node/9311 (Stand: 01.08.08)

3.6 Stolpersteine auf dem Weg zum Semantic Web Content Management

Eines der größten Probleme, das beim Import und Export von XML Dokumenten auftreten könnte, ist der generische Ansatz bei der Erstellung semantischer Beschreibungen [Diestelkamp, 2005]. Obwohl es möglich ist, unterschiedliche RDF Schemen zu kombinieren, existieren keine standardisierten Konventionen und damit die Gefahr, dass immer mehr unterschiedliche Beschreibungen zu einem unkontrollierbaren Wildwuchs führen. Aus diesem Grund haben sich im Internet einige Initiativen gegründet um bei der Verbreitung von allgemeinen Konventionen zur Beschreibung von Dokumenten Regeln zur Verfügung zu stellen. Die Initiative Dublin Core[28] arbeitet bereits seit 1994 an der Verbreitung von Standards für Metainformation. Derartige defacto Standards sind zwar nicht von offizieller Seite als Industriestandard anerkannt, können jedoch durch häufigen Gebrauch in der Alltagswelt wie ein echter Standard angesehen werden. Derart definierte Regeln können von RDF problemlos umgesetzt werden, schnüren jedoch die Freiheit bei der Beschreibung von Dokumenten ein. Individuelle Beschreibungen hingegen, stören die Austauschbarkeit von Informationen und erschweren das wofür sie eigentlich gedacht sind. Womöglich muss der richtige Weg für das semantische Web noch gefunden werden. Aber vielleicht könnten auch einheitliche Regeln durch die Verwendung von CMS dazu beitragen.

Ein weiteres Risiko ist, dass Metainformation heute hauptsächlich nur aus dem Grund der optimalen Platzierung von Seiten in den Ranglisten von Suchmaschinen benutzt wird. Daher könnte dem semantischem Web ein ähnliches Schicksal drohen, wenn semantische Information aus kapitalistischen Gründen zweckentfremdet und für sinnvolle Suchen unverwendbar wird. Daher könnte die Zukunft von semantischem Web Content Management womöglich in geschlossenen Wissensplattformen liegen, die das Konzept in einem kleinen aber konfliktfreien Bereich umsetzen. In solchen Umgebungen könnte die Technologie dazu dienen, Inhalte in Form eines Netzes zu strukturieren oder ein semantisches Glossar aufzubauen. Dann hätte diese Technologie aber nur Sinn, wenn der beschriebene Content groß genug ist, um diese Art der Organisation zu erfordern [Diestelkamp, 2005].

[28] http://dublincore.org/ (Stand: 01.08.08)

3.7 Forschungsfragen

Damit Systeme in der Lage sind, semantische Dienste auszuführen, ist ein gewisses technisches Grundgerüst notwendig. Das System muss RDF Informationen integrieren und verarbeiten können. Ist dies der Fall, kann die Information über verschiedene Scripts an das Format angepasst werden. BenutzerInnen müsste die Möglichkeit vorliegen, diese Information individuell zusammenzustellen. Je mehr semantische Dienste ein Content Management System aufweist, desto leichter ist es selbstverständlich, darauf aufbauende Anwendungen zu implementieren. Auch wenn das System nur über die Minimalanforderungen eines XML Handlings verfügt, könnte sich das Konzept mit einem mehr oder weniger großem Aufwand umsetzen lassen. Im Idealfall stehen dem System bereits Schnittstellen zu externen semantischen Diensten oder systemtypische Ontologien zum Datenaustausch zur Verfügung. In vielen Systemen wird gerade aktiv an der Entwicklung solcher Dienste gearbeitet. In dieser Arbeit kann ein guter Querschnitt über die aktuelle Verfügbarkeit und Entwicklungsstufe semantischer Technologien in CMS gegeben werden.

Um die Kompatibilität mit den vorgestellten Konzepten untersuchen zu können, müssen konkrete Forschungsfragen formuliert werden:

1. Besitzt das System das minimale technologisches Grundgerüst, das erforderlich ist, um dezentral gespeicherte semantische Information verarbeiten zu können?
   a) Können fremde XML Daten importiert und verarbeitet werden?
   b) Können darüber hinaus RDF Daten importiert oder erstellt werden?
   c) Können BenutzerInnen Meta Information für individuelle Suchkriterien oder die Organisation von Information benutzen?
2. Besitzt das System Schnittstellen zu externen semantischen Diensten oder systemeigene semantische Schnittstellen zum Datenaustausch?
3. Welche Entwicklungsgruppen bestehen und an welchen Erweiterungen wird gearbeitet?

3.8 Analyse der Systeme

Da die Zahl bestehender WCMS sehr groß ist, ist es unabdingbar die Menge der zu untersuchenden Systeme einzugrenzen. Hierzu werden einige Systeme repräsentativ für gewisse Gruppen ausgewählt. Die Leistungsmerkmale eines Systems hängen vorwiegend auch von den Anwendungen ab, für die es konzipiert wurde. Neben klassischen redaktionellen Systemen gibt es bereits heute Plattformen mit stark sozialem Charakter. Statische Systeme reihen sich neben dynamischen, geschlossenen oder modular aufgebauten Programmkonzepten ein. Dabei beschränkt sich diese Arbeit auf lizenzfrei verfügbare, also Open Source Produkte.

Zur Eingrenzung wird zunächst die Web Information Company Alexa[29] herangezogen, die statistische Auswertungen über Zugriffszahlen im Internet bereitstellt. Tabelle 3.1 enthält die Zugriffszahlen verschiedener CMS im April 2008[30].

Während die populären Systeme WordPress, Joomla! und Drupal bereits seit mehreren Jahren an der Spitze dieser Liste stehen, begannen die Zugriffszahlen auf klassische Systeme wie Typo3 zu sinken. Der von PacktPub[31] jährlich vergebene Open Source Content Management Award zeichnete 2007 Drupal, dahinter Joomla! als beste Open Source CMS aus. Als vielversprechendstes System wurde MODx[32] ausgezeichnet. WordPress und dahinter Drupal gewannen den Preis für das beste soziale CMS.

Um eine möglichst große Vielfalt unterschiedlicher Ansätze zu betrachten, wurde die Auswahl auf fünf Systeme eingegrenzt, die repräsentativ für ähnlich modellierte CMS herangezogen werden:

Zusammenfassung

- Typo3 (als klassisches redaktionelles CMS)
- Joomla! (als modernes redaktionelles CMS)
- Drupal (als Community basiertes CMS)
- MODx (als Komponenten basiertes Web 2.0 CMS)
- WordPress (als Weblog Publishing System)

[29] http://www.alexa.com (Stand: 29.06.08)
[30] http://www.blogsweek.com/en/most-popular-cms-in-2007/ (Stand: 01.08.08)
[31] http://www.packtpub.com/award (Stand: 29.06.08)
[32] http://www.modxcms.com (Stand: 01.08.08)

	OS CMS	**Rang**
1st	WordPress	619
2nd	Joomla!	850
3rd	Drupal	1196
4th	Pligg	10959
5th	ExpressionEngine	12732
6th	Mambo	17149
7th	MODx	19875
8th	Xoops	21312
9th	DotNetNuke	22233
10th	b2evolution	23155
11th	Plone	25035
12th	CMS Made Simple	25642
13th	eZ publish	27516
14th	Movable Type	30928
15th	Textpattern	36120
16th	PHP-Nuke	38795
17th	PHP-Fusion	45073
18th	e107	45514
19th	PostNuke	47302
25th	Typo3	58328

Tabelle 3.1: Vergleich aktueller Web Content Management Systeme aufgrund ihrer Zugriffsstatistik

Im Folgenden werden diese Systeme vorgestellt, um die Unterschiede zwischen ihnen näher zu beleuchten. Danach folgt eine Auswertung der gestellten Forschungsfragen. Hierzu wurden die Systeme lokal auf einem Apache HTTP Server[33] installiert und mit einer einfachen Beispielanwendung getestet. Da sich semantische Anwendungen nur über diverse erhältliche Erweiterungen integrieren lassen, wurde gleichzeitig die Verfügbarkeit von semantischen Erweiterungen untersucht und die aktuelle Arbeit zugehöriger Arbeitsgruppen beobachtet.

3.8.1 Typo 3

Das 1997 gestartete Projekt Typo3 sieht sich als Content Management Framework für kleine bis mittelgroße Unternehmen und verbindet eine große Palette an Funktionen, die durch Module erweitert und auch weiter entwickelt werden können. Das System wurde primär für AutorInnen geschaffen, um Inhalte auf einfache Weise zu generieren und mit nur wenigen Mausklicks zu publizieren. Für AdministratorInnen wird ein übersichtliches Rechtesystem bereitgestellt und für EntwicklerInnen stellt die strikte Trennung von Inhalt und Design eine Plattform zur optimalen Nutzung dar [Typo3, 2006]. In all diesen Punkten entspricht Typo3 dem Archetyp eines klassischen CMS, das aktiv entwickelt wird und zum Zeitpunkt dieser Arbeit in der Version 4.1.6 vorlag.

Über etwa 3000 verfügbare Plugins ist es möglich, das Standardgerüst des Systems zu erweitern. Hierbei wirkt sich die lange Entwicklungszeit und Popularität des Systems positiv aus, da sehr viele unterschiedliche Anwendungsfälle, reichend von Shop Systemen bis Wikis abgedeckt sind. Dennoch kommt die Ursprungsversion ohne jegliche Community Funktionalität aus und erfordert eine gewisse Einarbeitungszeit, da das System eine eigene Konfigurationssprache (TypoScript) verwendet, über die das System angepasst wird. Typo3 wurde als klassisches redaktionelles System ausgewählt, das aus der Zeit vor dem Web 2.0 stammt. Abbildung 3.3 zeigt die Startseite des CMS.

Technologische Merkmale:	
Scriptsprache	PHP 5
Datenbank	MySQL, PostgreSQL, Oracle
Erste Veröffentlichung	August 2000
Getestete Version	4.1.6 (März 2008)
Klasse	redaktionell strukturiertes CMS

[33] http://httpd.apache.org (Stand: 01.08.08)

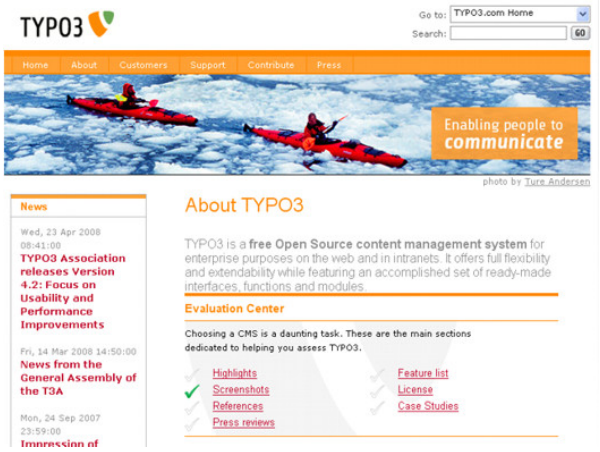

Abbildung 3.3: TYPO3 (http://www.typo3.org)

Auswertung

Die weite Verbreitung und große Zahl der Erweiterungen für Typo3 sorgt dafür, dass das System die technischen Voraussetzungen, die für semantische Dienste definiert wurden zumindest im Ansatz bereits mitbringt. So ist sowohl ein Export als auch ein Import von frei definierten XML Dateien möglich. RDF kann für RSS Feeds erzeugt und importiert werden. Auch das Erzeugen statischer Seiten ist möglich, damit Metadaten ihre Gültigkeit behalten. Zur Umsetzung der beschriebenen semantischen Dienste fehlt jedoch eine Importmöglichkeit von RDF Dateien und auch das Erstellen von RDF Schemen wird nicht unterstützt. Dies würde sich jedoch durch verschiedene Scripts bewerkstelligen lassen. Der Schritt zur Wissensplattform, in der BenutzerInnen Information individuell organisieren können, scheint jedoch noch weit entfernt, sofern überhaupt Bestrebungen in dieser Richtung bestehen.

Zusammenfassung

- Das Grundgerüst für XML Import/Export ist vorhanden.
- Es existiert ein Triplify Plugin, jedoch keine systemeigene Ontologie.
- Es wird keine Arbeitsgruppe für das Semantic Web unterstützt.

3.8.2 Joomla!

Das Joomla! Core Team, das das CMS aktiv entwickelt weist explizit darauf hin, dass das System insbesondere auf seinen Open Source Charakter und die damit verbundene Community, sowie die Einfachheit der Benutzung wert legt. Joomla! ist für BenutzerInnen ohne großes technische Verständnis gedacht, kann der Betreiberplattform nach aber auch für größere Unternehmen problemlos eingesetzt werden. Tatsächlich entspricht das Grundgerüst der Plattform eher einem redaktionellem WCMS und bildet den beschriebenen Publikations-Workflow ab. Dies verwundert nicht, da Joomla! eine Abspaltung des älteren Systems Mambo ist, und bei der ersten Veröffentlichung 2005 noch komplett auf der alten Software basierte. Die im Januar 2008 veröffentlichte Version 1.5. basierte hingegen auf einem komplett neu entwickelten Framework. Entgegen Typo3 setzt Joomla! besonders auf Simplizität und sieht seine Einsatzgebiete eher bei persönlichen Websites und kleineren Plattformen. Das System ist in Abbildung 3.4 zu sehen.

Für das CMS stehen viele Erweiterungen zur Verfügung, um das Produkt zu erweitern. Dies umfasst viele Community aber auch E-Commerce Elemente. Diese können über das Joomla! Extension Directory heruntergeladen werden[34]. Joomla! wurde als im Prinzip redaktionelles aber erweiterbares System deshalb ausgewählt, weil sein Ansatz als einfaches schnelles System ohne Einarbeitungszeit im starken Kontrast zu Typo3 steht und im Web 2.0 Zeitalter entstand. Insbesondere unterscheidet sich der Einsatzraum von Joomla! deutlich und orientiert sich mehr an privaten kleinen Projekten. Interessant ist, ob die Orientierung an unterschiedlichen Zielgruppen auch Auswirkungen auf die semantischen Entwicklungen innerhalb des Systems mit sich bringt.

Technologische Merkmale:	
Scriptsprache	PHP 5
Datenbank	MySQL
Erste Veröffentlichung	September 2005
Getestete Version	1.5.2 (März 2008)
Klasse	redaktionell strukturiertes CMS

Auswertung

Joomla! unterstützt zwar einfache RSS Funktionalität, das Grundgerüst für den Import von RDF Information fehlt aber noch. Selbst das Importieren einfacher XML Dateien

[34] http://extensions.joomla.org/ (Stand: 29.06.08)

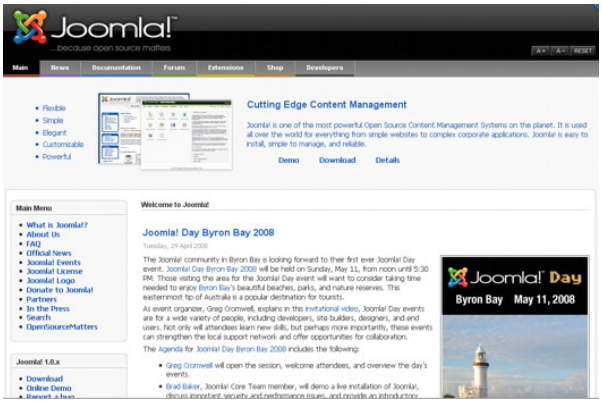

Abbildung 3.4: Joomla! (http://www.joomla.org)

wird problematisch, da es keine standardisierten Lösungen oder Erweiterungen gibt, um frei definierte XML Dateien zu verarbeiten. Die Implementierung semantischer Dienste scheint damit schwierig, da hier selbst die Grundfunktionalität erst geschaffen werden müsste. Jedoch gibt es Unternehmungen, das System langsam an das Semantic Web heranzuführen, wenngleich auch nicht durch interne Lösungen. Im März 2008 wurde das Projekt Code 04000* ins Leben gerufen, das eine Erweiterung durch die externe Software Triplify vorsieht. Damit setzt das System vorerst auf ein bereits in Entwicklung stehendes Projekt, um unterschiedliche Systeme miteinander zu verbinden. Zum Stand dieser Arbeit war das Projekt allerdings erst im Planungsstatus. Jedenfalls könnte es noch eine Weile dauern, ehe Joomla! für das semantische Web bereit ist.

Zusammenfassung

- Das Grundgerüst für XML Import/Export ist noch nicht vorhanden.
- Es gibt keine Schnittstellen zu externen Systemen.
- Es wird an einem Triplify Plugin gearbeitet.

3.8.3 Drupal

Drupal (siehe Abbildung 3.5) wird auf der Herstellerseite als Software angepriesen, die es individuellen BenutzerInnen oder Communities erlaubt, Webseiten auf sehr einfache Weise zu publizieren. Dabei zielt das Paket in erster Linie auf Community Aspekte, wie Diskussionsforen, Community Web Plattformen und Groupware ab. Als Stärke des

Systems wird wie auch bei Joomla! die Einfachheit der Bedienung hervorgehoben. Das Paket ist nach der Installation sofort lauffähig und einsetzbar. Drupal unterscheidet sich von klassischen Management Systemen deutlich und ist von den Prinzipien des Web 2.0 gezeichnet. Im Vordergrund stehen Community, Sharing und Tagging. Die installierte Plattform gleicht zunächst beinahe einem Wiki. Inhalte werden als Nodes gespeichert und können mit Tags versehen oder abonniert werden, um die Änderungen der Seite im Lauf der Zeit zu verfolgen. Hierin sieht man auch das Prinzip des Systems. BenutzerInnen sollen die Elemente des Projekts gemeinsam gestalten und entwickeln, das System ist weniger als Publikationswerkzeug, sondern als Organisator gedacht, der diese Inhalte geordnet zusammenführt [Kerres, 2006].

Daneben lässt sich Drupal auch für persönliche Websites und ferner für klassisches Content Management einsetzen. Das Grundgerüst ist durch Module erweiterbar und lässt sich auf verschiedene Anwendungsfälle anpassen. Ein interessanter Anwendungsfall von Drupal ist die Nutzung als Resource Directory. Hierbei kann wie bei einem Lexika oder Wiki ein gemeinsames Projekt erstellt und wie ein interaktives Buch entwickelt werden indem die einzelnen Nodes aufeinander verweisen. Da Drupal viel stärker auf neue Inhalte im Web ausgelegt ist, wird der Kontrast zu klassischen Systemen besonders interessant.

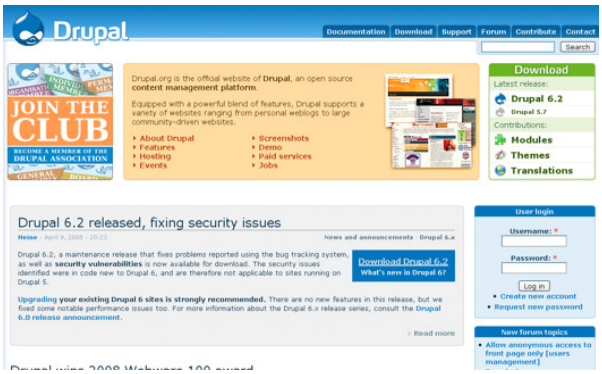

Abbildung 3.5: Drupal (http://www.drupal.org)

Auswertung

Aufgrund des Community basierten Ansatzes des Systems ist Drupal als Organisator verteilter Information konzipiert und nicht mehr weit vom semantischen Content Management entfernt. Drupal verfügt neben weitreichender RSS Funktionalität über zahlrei-

Technologische Merkmale:	
Scriptsprache	PHP 5
Datenbank	MySQL
Erste Veröffentlichung	Januar 2001
Getestete Version	6.2 (April 2008)
Klasse	Community basiertes CMS

che Import und Export Möglichkeiten von XML Dateien[35]. Für RDF Daten existiert eine offizielle RDF API[36] die eine RDF Datenbank, den Import und Export, sowie ein eigenes RDF Schema zur Verfügung stellt. Damit sind die Grundlagen für weitreichende semantische Dienste gegeben. Sowohl die Integration von dezentral vorliegender Information, als auch die Erstellung eigener RDF Schemen scheinen über mehr oder weniger einfache Scripts realisierbar. Für viele Anwendungsfälle existieren sogar bereits Lösungen, die in der Plugin Datenbank angeboten werden, etwa ein Modul für semantische Suche innerhalb des Systems. Das System könnte auf dem richtigen Weg sein, eine semantischen Wissensplattform zu werden.

Drupal unterstützt eine eigene Semantic Web Arbeitsgruppe[37], die seit 2007 an der Implementierung semantischer Dienste arbeitet. Das Projekt begeht damit so etwas wie eine Vorreiterrolle was die Verbreitung des Semantic Web im Content Management betrifft. Dies verwundert nicht, da bei der Programmierung bereits großen Wert auf die RSS Kompatibilität gelegt wurde und Drupal auf eine untypische Node basierte Struktur setzt, die die Voraussetzung für semantische Funktionalität mitbringt. Es bleibt abzuwarten, inwieweit sich semantisches Web in Drupal entwickelt und welche Rolle es in Zukunft spielen wird, aufgrund der Verfügbarkeit kann man das System aber getrost als die Semantic Web Plattform im CMS Bereich bezeichnen.

Zusammenfassung

- Das Grundgerüst für XML Import/Export ist vorhanden, es gibt eine eigene RDF API.
- Es existieren Plugins für Triplify und SIOC, sowie eine systemeigene Ontologie.
- Es wird eine Arbeitsgruppe für das Semantic Web unterstützt die aktiv an Erweiterungen arbeitet.

[35] http://basement.greenash.net.au/soc2006/ExistingImportExportModules (Stand: 29.06.08)
[36] http://drupal.org/node/222788 (Stand: 29.06.08)
[37] http://groups.drupal.org/semantic-web (Stand: 29.06.08)

3.8.4 MODx

MODx[38] (siehe Abbildung 3.6) ist ein relativ Junges System, das zur Zeit dieser Arbeit noch in der Beta Version vorlag, aber bereits 2007 den PacktPub Award für das vielversprechendste System des Jahres erhielt. MODx wurde im Oktober 2006 ins Leben gerufen und setzt die aktuellsten im Internet verfügbaren Technologien ein. So bezeichnet sich das Projekt selbst als das Ajax CMS von heute und morgen und integriert neben XHTML und RSS alle wesentlichen Elemente des Web 2.0. Das System verspricht komplette Meta Tag Kontrolle und integriert Dienste wie Ditto oder Ajax Suche. Dabei wurde vor allem darauf Wert gelegt, den Programmkern schlank zu halten. Zusätzliche Features sind als Komponenten verfügbar und lassen das System individuell zusammensetzen.

Es ist zu beachten, dass sich MODx an durchaus erfahrene BenutzerInnen richtet und ein gewisses Verständnis für die zugrunde liegenden Technologien voraussetzt. Dafür ist das Programm äußerst flexibel und darauf ausgelegt, alle Schranken klassischer Content Management Systeme zu öffnen. Wenngleich MODx ein anspruchsvolleres System darstellt, ist der Bezug zu recht jungen Technologien im Internet interessant und stellt die Frage ob MODx tatsächlich eine neue Generation von Content Management Systemen einläuten kann, wie es die Website des Projekts verspricht.

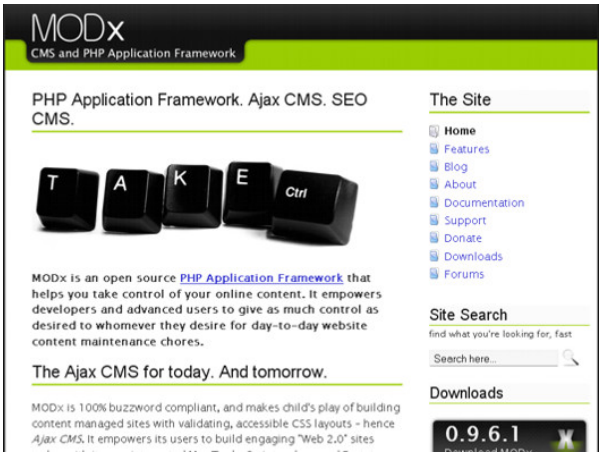

Abbildung 3.6: MODx (http://www.modx.com)

[38] http://modxcms.com/about.html (Stand: 29.06.08)

Technologische Merkmale:	
Scriptsprache	PHP 5
Datenbank	MySQL
Erste Veröffentlichung	-
Getestete Version	0.9.6.1 (November 2007)
Klasse	Komponenten basiertes Web 2.0 CMS

Auswertung

MODx besticht durch einen sehr übersichtlichen Kern, der durch Ajax interaktiv und nahe an Desktop Applikationen gehalten ist. Jede angelegte Seite kann durch eigene Meta Tags angereichert werden, um den Dokumentkopf für Suchmaschinen zu optimieren. Über Snippets können auch Teilelemente einer Seite automatisiert mit Meta Tags beschrieben werden. Die versprochene Meta Tag Kontrolle bezieht sich jedoch in erster Linie auf die Suchmaschinenplatzierung. Die einzelnen Seiten werden hier anders als bei den meisten CMS als statische HTML Dateien angelegt, was für den Export von langfristig gültigen Daten essentiell ist. Durch die über ein Plugin integrierbare Software Ditto[39] ist es außerdem möglich, Web 2.0 fähige Artikel, Blogs oder Foren zu erstellen. Hiermit können einzelne Artikel mit Tags versehen und nach Stichworten organisiert werden, wie es in vielen Web 2.0 Plattformen üblich ist. Darüber hinaus unterstützt Ditto den XML oder RSS Export von Seiten.

Eine Erweiterung um RDF oder XML Daten zu importieren existiert derzeit noch nicht. Obwohl das System stark den sozialen Charakter des Web 2.0 verfolgt und unterschiedliche Organisationsformen erlaubt, sind in der aktuellen Version noch keine semantischen Dienste geplant. Auch Arbeitsgruppen zu diesem Thema existieren zum Zeitpunkt dieser Arbeit noch nicht. Durch den modularen Charakter des Systems sind die Voraussetzungen für den Einbau semantischer Erweiterungen gegeben, deren Integration ist durch fehlende Plugins derzeit jedoch wahrscheinlich mit einem hohen Aufwand verbunden.

Zusammenfassung

- Das Grundgerüst für XML Import/Export ist in Form eines Plugins vorhanden.
- Es gibt keine Schnittstellen zu externen Systemen.
- Es wird keine Arbeitsgruppe für das Semantic Web unterstützt.

[39] http://ditto.modxcms.com/ (Stand: 29.06.08)

3.8.5 WordPress

Genau genommen ist WordPress (siehe Abbildung 3.7) kein CMS im eigentlichen Sinne, sondern ein Weblog Publishing System für einzelne BenutzerInnen, es verfügt aber seit 2005 auch über Content Management Funktionalität. WordPress existiert neben vielen großen Online Angeboten für private Weblogs als größtes Selfhosting System, das selbst installiert und angepasst werden kann. Es ist der Nachfolger des populären Systems b2[40] und zählt - nach der Alexa Zugriffsstatistik - zu den beliebtesten Systemen, die es aktuell am Markt gibt.

Das System zählt jedoch gemessen an der Funktionalität immer noch zu den schlankeren CMS, besitzt aber eine Bibliothek an erweiterbaren Elementen[41] um das System aufzurüsten. Die Kernelemente betreffen die Funktionalitäten eines Weblogs, also des Publizierens von Inhalten. Daneben enthält das System aber auch Benutzerkonten, Themen, XML Import oder RSS. Hinter WordPress steht eine aktive Community, die das Programm gemeinsam weiterentwickelt und verbessert. Die Entwicklung wie auch das System selbst folgt also den Web 2.0 Prinzipien.

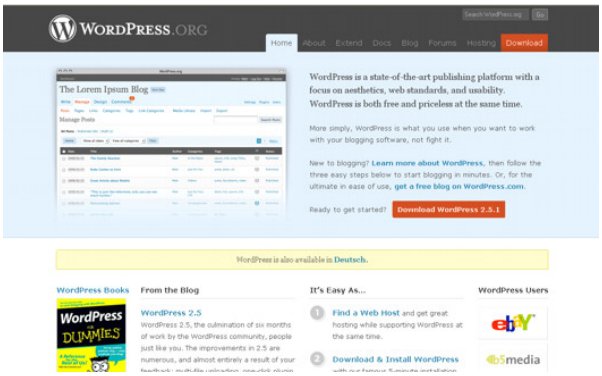

Abbildung 3.7: WordPress (http://www.wordpress.org)

Auswertung

Als schlankes Weblog Publishing System wartet die Grundinstallation des Systems lediglich mit Elementen eines redaktionellen Content Management Systems auf und unterstützt den Import von Daten, die über sogenannte Importers direkt von den Systemen

[40] http://cafelog.com/ (Stand: 15.09.08)
[41] http://wordpress.org/extend/plugins/ (Stand: 29.06.08)

Technologische Merkmale:	
Scriptsprache	PHP 5
Datenbank	MySQL
Erste Veröffentlichung	Januar 2004
Getestete Version	2.5 (März 2008)
Klasse	Weblog Publishing System

Movable Type[42], Textpattern[43], Greymatter[44], Blogger[45] und b2 übernommen werden können. Damit ist zumindest ein breites Feld ähnlicher Applikationen abgedeckt. Für den Austausch zwischen unterschiedlichen WordPress Installationen kann die gesamte Information als XML Datei exportiert und importiert werden. Jedoch geschieht dies über einen Manager und erlaubt keine individuellen Einschränkungen. Dem System fehlt eine flexible XML Verarbeitung, um RDF Schemen übernehmen zu können. Hier macht sich bemerkbar dass bei diesem System auf einen kleinen Kern für einfache Anwendungen wert gelegt wurde.

Obwohl es nicht im offiziellen Plugin Directory angeboten wird, existieren für WordPress Erweiterungen, um SIOC kompatible Dateien zu erzeugen beziehungsweise zu importieren[46]. Damit können Daten nach einer für Blogging Systeme konzipierten Ontologie ausgetauscht werden. Es verwundert nicht, dass Systeme wie WordPress nicht an eigenen Lösungen für semantische Dienste arbeiten, sondern auf externe Gruppen zugreifen, die versuchen einen einheitlichen Standard zu schaffen. Da die Breite an bestehenden Blogging und Foren Systemen extrem groß ist, kann auf diese Weise erreicht werden, dass Systeme über eine Brücke miteinander kommunizieren können. Dies ermöglicht einen einfachen Datenaustausch, von semantischer Organisation von Information oder E-Commerce Lösungen ist das System jedoch weit entfernt. Dies liegt wahrscheinlich auch nicht im Sinn der Applikation, die nicht als CMS konzipiert wurde.

Zusammenfassung

- Das Grundgerüst für XML Import/Export ist vorhanden.
- Es existiert ein SIOC Plugin, zudem eine systemeigene Ontologie.
- Es wird an semantischen Erweiterungen gearbeitet.

[42] http://www.movabletype.org/ (Stand: 15.09.08)
[43] http://textpattern.com/ (Stand: 15.09.08)
[44] http://www.noahgrey.com/greysoft/ (Stand: 15.09.08)
[45] https://www.blogger.com/start (Stand: 15.09.08)
[46] http://sioc-project.org/wordpress/ (Stand: 29.06.08)

3.9 Diskussion

Die Analyse zeigt, dass noch große Unterschiede bei der Bereitstellung semantischer Schnittstellen bestehen. Das Ziel einer kompletten Vernetzung unterschiedlicher Systeme liegt daher noch in Ferne. Viele Systeme setzen auf externe Software Lösungen um Informationen zwischen unterschiedlichen Plattformen auszutauschen. Die Projektgruppen Triplify und SIOC bieten solche Schnittstellen an, die Ontologien für CMS zur Verfügung stellen. Obwohl es bisher keine offiziellen Standards für den semantischen Datenaustausch gibt, besitzt SIOC bereits eine W3C Kandidatur Status und könnte sich bald etablieren. Interne Lösungen besitzen hingegen den Vorteil, dass sie exakt an ein System angepasst werden können. Daher besteht der Anreiz systemeigener Entwicklungen in einer einfacheren Handhabung, speziell für technisch uninteressierte BenutzerInnen. An der Integration dieser Dienste wird in vielen CMS zur Zeit gearbeitet.

Die Integration externer semantischer Daten gestaltet sich dennoch je nach System noch mehr oder weniger schwierig. Da oftmals selbst die Grundfunktionalität wie eine freie XML Gestaltung noch fehlt, bedarf es einiger Entwicklungsarbeit, um die Kompatibilität mit RDF zu gewährleisten. Das Ziel einer semiautomatischen Inhaltserstellung durch externe Daten ist realisierbar, jedoch zum jetzigen Zeitpunkt quasi nicht vorhanden. Bei der Wahl eines CMS muss daher insbesondere darauf geachtet werden, dass zumindest eine weitläufige XML Unterstützung oder sogar eine RDF Schnittstelle vorhanden ist.

Im Bereich semantischer Dienste, wie individuelle Organisation von Information oder semantische Suchen, zeigen sich die Systeme ebenfalls noch wenig gerüstet. CMS beginnen sich langsam an Dienste des Web 2.0 zu orientieren und setzen auf gemeinschaftliche Organisationsformen wie Tagging. Selbst moderne Systeme wie MODx haben das Potential von semantisch beschriebenen Daten als Organisationsform noch nicht erkannt. Eine Ausnahme stellt die Community Plattform Drupal dar, die eine eigene Arbeitsgruppe für semantische Dienste betreibt. Dies betrifft nicht nur eine freie Gestaltung von RDF Schemen oder den Import von semantischen Daten, sondern auch Plugins für semantische Suchen. Insgesamt kann man davon ausgehen, dass es noch einige Zeit in Anspruch nehmen wird, bis die Idee des Semantic Web vollends im Bereich des Content Management Einzug findet. Der erste Schritt zu einem semantisch organisierten Web ist in jedem Fall bereits getan.

4 Praktischer Teil - Grundlagen

Dieses Kapitel enthält eine Einführung in jene Komponenten, die als Basis für die Implementierung einer semantischen Beispielanwendung dienen werden. Nach einer Einleitung und Darstellung der Motivation für eine semantische Schnittstelle im Content Management System Drupal wird insbesondere auf die Entwicklung innerhalb des Systems eingegangen. Ebenso werden die zur Verfügung stehenden Programmierschnittstellen beleuchtet, die im Zuge der Implementierung verwendet werden. Danach wird das technologische Gerüst der semantischen Datenbank DBpedia besprochen, die als Quelle für semantische Abfragen genutzt werden soll.

4.1 Einleitung

Die ersten semantischen Anwendungen, die sich im Content Management etablieren, betreffen hauptsächlich den Datenaustausch zwischen Systemen gleicher BetreiberInnen, oder die Verarbeitung externer semantischer Informationen aus den bisher nur vereinzelt vorliegenden Quellen. Projekte wie Freebase[1], TrueKnowledge[2] und DBpedia[3] verfolgen diese Idee und versuchen große semantisch annotierte Datenbanken aufzubauen, auf die extern zugegriffen werden kann. Bei der Umsetzung eines Moduls für das Semantic Web Content Management können diese Datenquellen als Grundlage für eine semantisch basierte Integration von Daten dienen.

Unter den Open Source Systemen kann die Community Plattform Drupal als eine Art Pionier hervorgehoben werden, da das Projekt mehrere Bestrebungen verfolgt, semantische Dienste in naher Zukunft zugänglich zu machen. Neben Implementierungen von Schnittstellen zu SIOC und Triplify, wird derzeit aktiv an der Entwicklung einer RDF

[1] http://www.freebase.com/ (Stand: 15.09.08)
[2] http://www.trueknowledge.com/ (Stand: 15.09.08)
[3] http://DBpedia.org (Stand: 15.09.08)

API gearbeitet, für die es schon erste Anwendungen gibt [Drupal, 2008f]. Diese Anwendungen werden als Module zur Grundversion frei verfügbar gemacht und können auch individuell adaptiert werden. Das Modul Skype Status verwendet beispielsweise die RDF Information über den Anwesenheitsstatus eines Kontos des Internettelefonie-Anbieters Skype[4], um diesen in ein User Profil in Drupal zu integrieren. Das SPARQL Protocol and RDF Query Language (SPARQL)[5] Modul bietet eine Implementierung, um RDF Daten der API mittels Abfragen zu verarbeiten. Dennoch ist es in Drupal bisher nicht möglich, Inhalte aus externen semantischen Daten zu erzeugen.

Da es das Ziel dieser Arbeit ist, die Anforderungen an das Semantic Web Content Management in einer möglichst praxisnahen Anwendung umzusetzen, wird daher Drupal als Plattform für eine Anwendung ausgewählt, die das Content Management System, um eine Schnittstelle zu der semantischen Datenbank DBpedia erweitert. Dadurch soll es möglich werden, das Ziel semiautomatischer Erstellung von Inhalten auf Basis semantischer Daten im Content Management zu realisieren. Mit der Abfragesprache SPARQL sollen komplexe Zusammenhänge aus der Wissensdatenbank extrahiert und in den redaktionellen Prozess des Systems integriert werden. Damit wird ein wesentlicher Anwendungsfall des Semantic Content Management praktisch umgesetzt [Pellegrini, 2006].

Für die Realisierung dieses Konzepts wird ein Vorgehensmodell angewandt, das sich aus Anforderungsdefiniton, Analyse, Entwurf, Programmierung und Validierung zusammensetzt. Bevor eine Anforderungsanalyse und entsprechende Anwendungsszenarien erstellt werden können, ist es jedoch notwendig, die technischen Konzepte der beteiligten Komponenten zu beleuchten. Daher wird im Folgenden näher auf die Entwicklung in Drupal, und die Möglichkeiten der RDF und SPARQL API des Systems eingegangen. Danach erfolgt eine Darstellung der Datenstruktur der Wissensdatenbank DBpedia, die als Inhaltsquelle für semantische Abfragen dienen wird.

4.2 Entwicklung in Drupal

In diesem Abschnitt wird die technische Struktur des Content Management Systems Drupal vorgestellt, dessen Basisversion um eine Schnittstelle zu der semantischen Plattform DBpedia erweitert werden soll. Daher wird in erster Linie auf die Möglichkeiten der Entwicklung von eigenen Modulen eingegangen. Eine gute Einführung in die Arbeit mit dem System aus AnwenderInnensicht findet sich in [Graf, 2006]. Um die Struktur des

[4] http://www.skype.com (Stand: (01.10.08)
[5] http://www.w3.org/TR/rdf-sparql-query/ (Stand: 15.09.08)

Systems zu verstehen, müssen erst einige Kernkonzepte beleuchtet werden, auf denen Drupal aufbaut.

Front- und Backend: Drupal verwendet (wie die meisten CMS) ein Frontend, das die Inhalte auf BesucherInnen zugeschnitten als Webseite präsentiert. Daneben können alle Inhalte und Prozessabläufe auf einer Verwaltungsoberfläche, dem Backend, angezeigt werden.

Datenbank: Die Inhalte, also Text, Bilder, Links oder Kategorien werden dynamisch aus einer Datenbank ausgelesen, die Tabellen und Zeilen verwendet. Jede Tabellenzeile setzt sich aus Feldern zusammen, in denen die tatsächlichen Daten enthalten sind. Dadurch kann das Layout und der Inhalt der Seite getrennt werden.

Inhalte: Drupal bietet unterschiedliche Typen von Inhalten wie Blogs oder Stories. Das Erscheinungsbild und die angezeigten Daten sowie Zugriffsrechte sind für jeden Inhaltstyp einstellbar.

Zugriffsrechte: Zugriffsrechte sind ein wichtiges Konzept, das dazu dient, Ressourcen zweckdienlich einzusetzen. Da unterschiedliche Benutzergruppen unterschiedliche Inhalte bearbeiten oder einsehen dürfen, werden nur entsprechende Seitenelemente angezeigt und aus der Datenbank ausgelesen. Auf diese Weise werden auch die Ladezeiten verkürzt.

Templates: Templates sind visuelle Schablonen, die über den Inhalt einer Seite gelegt werden. Durch sie werden unterschiedliche Seitenattribute wie Schriftgröße, Farbe und Layout definiert. Um das Layout der Webseite zu ändern, muss nur die jeweilige Schablone gewechselt werden. Drupal verwendet *Themes* in denen derartige Schablonen enthalten sind. Solche Themes können im Internet heruntergeladen oder selbst erstellt werden.

Module: Durch Module kann die Kernversion des Systems individuell erweitert werden. Drupal verfügt über zahlreiche vorgefertigte Module wie Online Shops oder Forensysteme. Module, die auf ein Problem zugeschnitten sind, können aber auch selbst entwickelt werden, um spezifische Anforderungen zu erfüllen.

Konfiguration: Die Konfiguration bestimmt, wie die Seite strukturiert und angezeigt wird und welche Benutzergruppen Inhalte einsehen dürfen. Die Konfiguration erlaubt es, die Plattform individuell zu gestalten ohne in den Quelltext eingreifen zu müssen.

4.2.1 Nodes

Bei der Strukturierung und Organisation des gesamten Inhaltes einer Drupal Installation werden zwei wesentliche Konzepte verwendet: Nodes und Blöcke. Eine Node (siehe [Drupal, 2008c]) ist ein beliebiger Inhaltstyp und kann gewissermaßen mit einem Baustein verglichen werden. Nodes können ein Teil eines Forums, Blogs oder Buches sein. Auch die Definition eigener Nodes ist möglich. Dabei muss jede Node einen Typ besitzen, der ihren Inhaltstyp beschreibt. Darüber hinaus besitzt sie eine ID, einen Titel, einen Inhaltsteil, einen Datumsstempel, einen Autor und eventuell weitere Eigenschaften. Alle diese Daten werden in einer Tabelle der Datenbank abgelegt. Außerdem enthält jede Node eine unlimitierte Anzahl an Kommentaren, die zu jedem beliebigen Inhaltstyp abgegeben werden können. Die Adressierung einer Seite des Systems erfolgt über die ID der jeweiligen Node, die URL *http://drupal.org/node/1* referenziert beispielsweise die Node mit der ID 1.

Der Aufruf einer bestimmten URL über die Adresszeile oder einen Menülink entspricht einer Datenbankabfrage mit der Anweisung, bestimmte im Node Typ definierte Tabelleninhalte auszulesen. Derartige Datenbankabfragen sind in Modulen implementiert. Die Nodes werden daraufhin durchsucht und ein nach Datum sortiertes Ergebnis angezeigt. Die Anzeige richtet sich auch nach den zugeteilten Zugriffsrechten, die über die Anzeige von unterschiedlichen Tabelleneinträgen entscheiden.

In der Kernversion enthält Drupal folgende Node- oder Inhaltstypen [Drupal, 2008d]:

Blog Eintrag: Blogs oder Weblogs dienen als persönliches Tagebuch und können zur Veröffentlichung von eigenen Gedanken verwendet werden.

Buchseite: Drupal erlaubt die Erstellung gemeinsamer Bücher über bestimmte Themen. Zu diesem Buch können über Buchseiten Kapitel verfasst werden. Das Konzept ähnelt einem Wiki[6].

Kommentar: Kommentare werden zu Nodes abgegeben und stellen einen Inhaltstyp, aber keine eigene Node dar.

Forum: Ein Forum kann in Drupal als Set von Nodes und ihren Kommentaren aufgefasst werden. Die Gruppierung der Nodes zu einem Forum erfolgt über die Taxonomien, die Nodes einem bestimmten Thema zuordnen. Solche Taxonomien können frei erstellt werden.

[6] http://de.wikipedia.org/wiki/Wiki (Stand: 01.10.08)

Seite: Einfache (statische) Seiten können ebenfalls über eine Node realisiert werden.

Abstimmung: Abstimmungen beinhalten ein Formular und erlauben über ein bestimmtes Thema zu diskutieren und eine Stimme abzugeben.

Story: Dieser Node Typ entspricht dem üblichen Artikel eines CMS. Stories können mit einem bestimmten Ablaufdatum versehen werden, das über die Anzeige entscheidet.

4.2.2 Blöcke und Boxen

Die Struktur der jeweils angezeigte Seite setzt sich aus Blöcken und Boxen zusammen, in denen die Datenbankinhalte angezeigt werden. Blöcke können beliebige Elemente beinhalten, etwa ein Menüsystem, Statusprotokolle oder aktuelle Meldungen. Die Positionierung der Blöcke kann frei gewählt werden. Die Inhaltstypen werden in Boxen angezeigt. Abbildung 4.1 zeigt diese Elemente innerhalb einer typischen Drupal Seite.

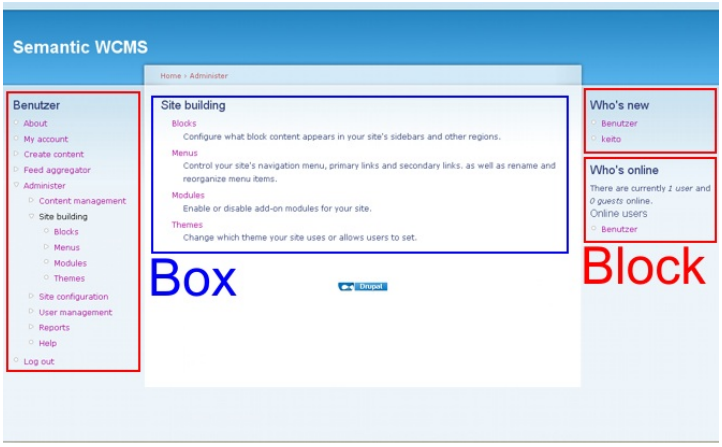

Abbildung 4.1: Blöcke und Boxen in Drupal

4.2.3 Module und Hooks

Das Drupal Modul System basiert auf sogenannten *Hooks* (*to hook in* bedeutet *einhängen*) durch die mit dem Kerngerüst interagiert werden kann [Drupal, 2008a]. Über diese

Programmierschnittstellen kann jede Aktion implementiert werden ohne in bestehenden Quelltext einzugreifen. Ein Hook ist eine PHP Funktion, die sich aus dem Namen des Moduls und dem Namen des Hooks zusammensetzt. Die Funktion *example_help()* etwa bezeichnet die Implementierung des *help* Hooks im Modul *example*. Jede dieser Funktionen hat ein definiertes Set von Parametern und Rückgabewerten. Das System greift auf die Implementierung eines eigenen Hooks zu, indem alle Hooks eines jeweiligen Typs aufgerufen werden. Eine Übersicht über die zur Verfügung stehenden Hook Typen befindet sich in Anhang 9.1.

Um dieses System zu verdeutlichen, wird nun die Implementierung einer einfachen Erweiterung über ein Modul vorgestellt. Dieses Modul soll einen neuen Block Typ zur Verfügung stellen, in dem alle BenutzerInnen der Seite aufgelistet werden, die am aktuellen Tag Geburtstag haben. Hierzu wurde zunächst das Zusatzmodul Profil aktiviert und ein neuer Datentyp *geb_datum* definiert, der bei der Registrierung die Datumseingabe des Geburtstages erfordert. Der entsprechende Wert wird in der Datenbank zusätzlich zu den BenutzerInnendaten abgelegt.

Das Zusatzmodul wird als neuer Ordner *birthday* im Verzeichnis *modules* der Drupal Installation angelegt und besteht aus zwei Grunddateien:

birthday.info Die *.info Datei enthält grundlegende Informationen, die vom System ausgelesen und später im Backend angezeigt werden. Sie dienen der Beschreibung des Moduls.

birthday.module Der eigentliche Inhalt des Moduls, sowie die Kommunikation mit dem Programmkern werden in einer *.module Datei geschrieben, die auch auf andere Dateien zurückgreifen kann.

Die beiden Komponenten werden für dieses Beispiel nun im Detail vorgestellt.

Info Datei

Listing 4.1 zeigt den Aufbau der *birthday.info* Datei. Die Notation folgt den Konventionen des INI Standards[7] für Konfigurationsdateien, damit sie von der PHP Funktion *parse_ini_file()*[8] ausgelesen werden kann. Sie beginnt mit einem *;$Id:[Titel]$* Eintrag, der die Datei identifiziert und der Projektentwicklung dient. Auf Drupal ist es geläufig,

[7] http://en.wikipedia.org/wiki/Ini_file (Stand: 21.07.08)
[8] http://at2.php.net/parse_ini_file (Stand: 21.07.08)

das Module als offene Projekte angelegt und verteilt entwickelt werden. Danach folgen einige Felder, wobei eine Namensgebung (*name*), eine Beschreibung (*description*) und eine Versionszugehörigkeit (*core*) zwingend sind. Daneben kann die Datei zusätzliche Konfigurationselemente oder Paketabhängigkeiten besitzen. Diese Informationen werden von Drupal verwendet, um das Modul zu identifizieren und im Administrationsbereich anzuzeigen. Hierzu muss zunächst einfach eine *.module Datei angelegt und ebenfalls mit einer ID Bezeichnung versehen werden. Als Besonderheit benötigen Drupal Modul Dateien keinen PHP Closing Tag und verwenden lediglich *<?php* statt des üblichen Tag Paars *<?php ?>*. In Abbildung 4.2 ist das neue Modul im Backend von Drupal zu sehen.

Listing 4.1: Birthday.info

```
1 ;$Id: birthday.info, Semantic Web Content Management$
2 name = birthday
3 description = Zeigt einen Block an, in dem allen Geburtstagskindern gratuliert wird.
4 datestamp = "1216377383"
5 core = 6.x
```

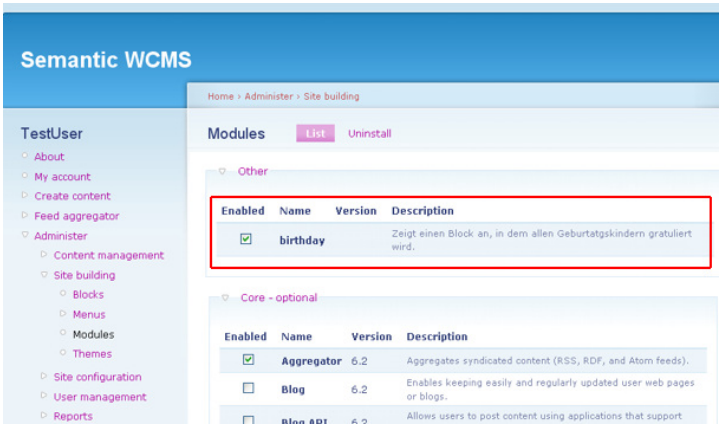

Abbildung 4.2: Das Modul im Administrationsbereich von Drupal

Module Datei

Die *.module Datei besteht neben einem ID Eintrag aus der Implementierung mehrerer Hooks, mit denen die Kommunikation mit dem Programmkern realisiert wird. Im Prinzip kann an dieser Stelle jeder Hook implementiert werden. Der Aufruf erfolgt vom Programmkern durch Listener, die überprüfen, ob ein Hook vorhanden ist. Ein Modul

sollte zumindest die Implementierung der Schnittstellen *hook_perm()*[9] zur Zugriffskontrolle und *hook_help()*[10] für die Funktionsbeschreibung enthalten. Die Zugriffskontrolle wird hier nur durch einen allgemeinen Eintrag dargestellt, der dem System lediglich mitteilt, dass eine solche Kontrolle vorhanden ist (Siehe Listing 4.2).

Listing 4.2: hook_perm() in birthday.module

```
1 <?php
2   // Zugriffsrechte definieren
3   function birthday_perm() {
4     return array('access birthday content');
5   }
6 ?>
```

Da in diesem Beispiel ein neuer Block entwickelt werden soll, wird zudem der hook *hook_block()*[11] benötigt, in dem ein Datenbankzugriff erfolgt, um die Namen der BenutzerInnen zu ermitteln, die am aktuellen Tag Geburtstag haben. Auch die Formatierung der Ausgabe wird in diesem Abschnitt implementiert. Listing 4.3 enthält den Aufbau des *hook_block()* Hooks. Die Variable *$op* unterscheidet die Darstellungsart des Blocks. So wird in *$op="list"* die Listendarstellung im Administrationsbereich festgelegt, in *$op="view"* steht der für BenutzerInnen sichtbare Inhalt. Dieser wird über ein Array *$block[ ]* an das System zurückgegeben. Die Variable *$block_content* enthält die formatierte Anzeige der Ausgabe, die in HTML Form als String angegeben wird.

Listing 4.3: hook_block()

```
7  <?php
8    function birthday_block($op='list', $delta=0) {
9      if ($op == "list") {
10       $block[0]["info"] = 'Birthday Block';
11       return $block;
12     }
13     else if ($op == 'view') {
14       $block_content = 'Inhalt';
15     }
16     // Set Up des Blocks
17     $block['subject'] = 'Birthday Block';
18     return \$block;
19     }
20   }
21 ?>
```

[9] http://api.drupal.org/api/function/hook_perm/6 (Stand: 21.07.08)
[10] http://api.drupal.org/api/function/hook_help/6 (Stand: 21.07.08)
[11] http://api.drupal.org/api/function/hook_block/6 (Stand: 21.07.08)

Um die Daten des Profil Feldes *geb_datum* zu erhalten, ist eine Datenbankabfrage erforderlich, die als SQL Abfrage in die Variable *$block_content* geschrieben wird. Sie verknüpft die Datenbanktabellen *profile_values* und *users*. Eine Verbindung und Trennung zur Datenbank ist nicht nötig, da das Modul vom Programmkern aufgerufen wird, der bereits über diese Verbindung verfügt. Aus dem Ergebnis wird HTML Code erzeugt, der den BenutzerInnennamen und einen Link zum jeweiligen BenutzerInnenkonto beinhaltet. Dieses wird in Listing 4.4 gezeigt. Der vollständige Inhalt der birthday.module Datei kann in Anhang 9.2 nachgeschlagen werden.

Listing 4.4: Inhalt des Blocks

```
22 <?php
23   $block_content = '<p>Geburtstagskinder:</p>';
24   $query="SELECT u.name, u.uid from profile_values p, users u
25           WHERE p.fid = 1 AND p.uid = u.uid AND p.value like '%";
26   $query.=date("md");
27   $query.="\%';";
28   $result = db\_query($query); // User Namen extrahieren
29
30   while ($users = db_fetch_object(\result)) {
31     $block\_content .= l($users->name, $users->uid) . '<br />';
32   }
33 ?>
```

In Abbildung 4.3 ist nun das System aus BenutzerInnensicht zu sehen, in dem der aktivierte Block enthalten ist. Er zeigt den angegebenen Testdatensatz als Link im rechten Seitenbereich an. Obwohl dieses Beispiel nur den konkreten Fall der Implementierung eines Blocks beleuchtet, kann das Konzept auch für andere Inhaltstypen übernommen werden. Die Implementierung einer eigenen Node erfordert zwar einigen Mehraufwand, besteht aber im Wesentlichen aus den gleichen Schritten. Dies wird später noch gezeigt.

4.3 Die RDF API

Nach Gründung der Semantic Web Group[12] im Jahr 2007 setzte es sich das Entwicklungsteam vom Drupal zum Ziel, so rasch wie möglich eine Anbindung des Systems zum Semantic Web zu schaffen. Neben Schnittstellen zu Triplify und SIOC, ist die Entwicklung der RDF API ein wesentlicher Schritt in diese Richtung. Obwohl die RDF API zum Zeitpunkt dieser Arbeit noch in der Alpha Version vorliegt, ist das Grundgerüst für

[12] http://groups.drupal.org/semantic-web (Stand: 01.10.08)

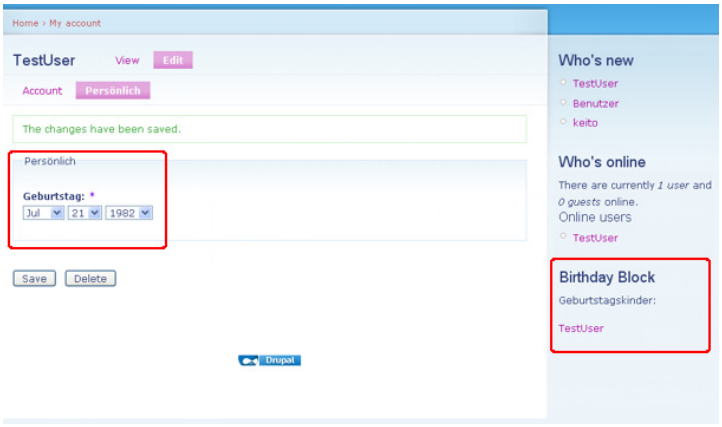

Abbildung 4.3: Anzeige des Birthday Blocks im Frontend von Drupal.

semantisches Arbeiten vorhanden. In diesem Abschnitt werden die Kernelemente dieser RDF API, die als Basis für eine eigene Entwicklung dienen wird, näher vorgestellt.

4.3.1 RDF API Konzepte

Die RDF API von Drupal ermöglicht die Definition und den Import/Export von RDF Aussagen, die über eigene Module abgefragt und bearbeitet werden können. Um dies zu realisieren, verwendet die API einige grundlegende Konzepte [Drupal, 2008e]:

Repositories

Ein Repository stellt eine Art Behälter für RDF Daten dar. Die Form der Implementierung dieses Behälters kann auf unterschiedliche Weise erfolgen. Beispielsweise als serialisierte Datei, innerhalb eines Datenbankmanagementsystems, speicherintern, oder auch als Remote Procedur Call (RPC)[13] Verbindung zu einem anderen System, wie mittels XML-RPC[14].

Es existieren bereits die Module *RDF DB*, das einen Datenbank basierten Behälter implementiert, und das *RDF Import Modul*, das einen beliebigen serialisierten RDF

[13] http://tools.ietf.org/html/rfc1831 (Stand: 15.09.08)
[14] http://www.xmlrpc.com/ (Stand: 23.07.08)

Graf als Auslesebehälter nutzt, beispielsweise eine lokal gespeicherte XML Datei. Die Definition eines Behälters geschieht über den Hook *hook_rdf_repositories()*.

Kontexte

Ein Kontext ist ein ungeordnetes Set von RDF Aussagen in Form eines benannten Grafs[15]. Die Idee benannter Grafen sieht die Zusammenfassung verschiedener RDF Grafen in einem Dokument vor, die durch URIs gekennzeichnet werden. Eine Syntaxsprache dieses Ansatzes ist TRIX[16].

Kontexte können als Beziehung zwischen RDF Aussagen und ihrem zugehörigen Teil Graf betrachtet werden. Durch Kontexte ist es möglich, die Herkunft einer RDF Aussage zu verfolgen. Dadurch können alte RDF Daten durch Neuere ersetzt werden. Sie werden durch den Hook *hook_rdf_contexts()* implementiert.

Namensraum

Um URI Bezeichnungen, wie beispielsweise die Dublin Core Spezifikation, abzukürzen, verwendet die Drupal RDF API Namensräume. Als Namensräume können beispielsweise die Syntaxsprachen CURI[17] und QName[18] verwendet werden, die jeweils vom W3C vorgeschlagen wurden. Um eine abgekürzte Syntaxsprache zu verwenden, muss ein gesamtes RDF Vokabular neu in dieser Sprache spezifiziert werden. Dies dient lediglich der besseren Lesbarkeit. Drupal Module können Namensräume mit dem Hook *hook_rdf_namespaces()* definieren.

Formats

Als Syntaxsprache für die eigentlichen RDF Aussagen werden die Formate RDF/PHP[19] und RDF/JSON[20] von der Drupal RDF API unterstützt. Das RDF/PHP Format behandelt RDF Aussagen als zwei dimensionale Arrays, die jeweils aus einem RDF Triple

[15] http://www.w3.org/2004/03/trix/ (Stand: 23.07.08)
[16] http://sw.nokia.com/trix/TriX.html (Stand: 23.07.08)
[17] http://www.w3.org/TR/curie/ (Stand: 23.07.08)
[18] http://www.w3.org/TR/REC-xml-names/#ns-qualnames (Stand: 23.07.08)
[19] http://drupal.org/node/219870 (Stand: 23.07.08)
[20] http://n2.talis.com/wiki/RDF_JSON_Specification (Stand: 23.07.08)

bestehen. RDF/JSON ist ein verwandter Ansatz, der auf einer möglichst einfachen maschinellen Verarbeitbarkeit basiert. Eine größere Interoperabilität mit anderen Formaten kann über die Import und Export Module der API erreicht werden, die auch andere Formate wie RDF/XML und N3 verarbeiten können. (Vgl. Abschnitt 2.3.2) Die Zahl der unterstützten Formate kann mit dem Hook *hook_rdf_formats()* erweitert werden.

4.3.2 RDF Funktionen

Die RDF API stellt eine Vielzahl an Funktionen für die Entwicklung eigener aufsetzender Module zur Verfügung. Eine Übersicht der PHP Funktionen der API befindet sich in [Drupal, 2008e].

4.3.3 RDF API Hooks

Als Programmierschnittstellen stehen fünf Hooks zur Verfügung, die eine vielschichtige Interaktion mit dem Grundgerüst erlauben [Drupal, 2008e]:

hook_rdf(): Aktionen vor und nach dem Einfügen einer RDF Aussage.

hook_rdf_repositories(): Spezifikation eigener Repositories.

hook_rdf_contexts(): Definition von Kontexten.

hook_rdf_namespaces(): Formulierung von Namensräumen.

hook_rdf_formats(): Definition von RDF Syntaxen.

4.4 DBpedia

Die Datenbank DBpedia nutzt eine vollständige Kopie der geparsten Daten der Wikipedia, um semantisch annotierte Inhalte zur Verfügung zu stellen. Kernkonzept der Anwendung ist, den derzeit bestehenden Mangel an verwertbaren RDF Daten durch eine Kopie der zur Zeit größten Wissensplattform und einer einheitlichen Ontologie entgegen zu wirken. Die Anwendung steht derzeit als Prototyp zur Verfügung, es wird aber intensiv an der Bereitstellung von unterschiedlichsten Zugangspunkten und Schnittstellen

gearbeitet.[21] Das Entwicklungsteam strebt insbesondere eine Vernetzung der Daten mit anderen semantischen Datenquellen an, die mit Semantic Web Browsern abgerufen werden können. Die Verlinkung funktioniert vergleichbar mit traditionellen Links in HTML Browsern, indem RDF Links zwischen Daten geknüpft werden. Auf diese Weise können auch Abfragen über verstreute Daten vorgenommen werden, wie es die Idee des Semantic Webs vorsieht [DBpedia, 2008]. Auf Abbildung 4.4 ist der Stand der Verlinkung von DBpedia mit anderen semantischen Plattformen zum Zeitpunkt der Veröffentlichung dieser Arbeit zu sehen. Hierbei stellen die blauen Knoten direkte, die grauen Knoten indirekte Verbindungen zu der semantischen Datenbank dar.

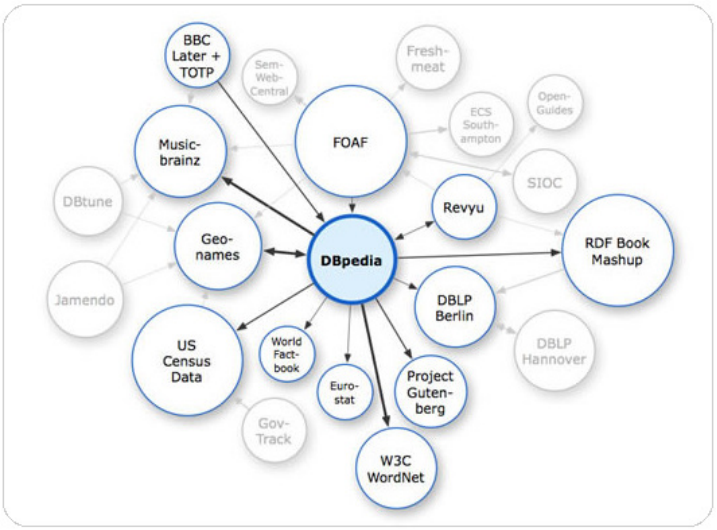

Abbildung 4.4: Schnittstellen zwischen semantischen Plattformen [DBpedia, 2008]

Die folgenden Abschnitte widmen sich dem technischen Konzept von DBpedia und zeigen, wie Daten aus der Datenbank extrahiert und semantisch verarbeitet werden können.

4.4.1 Datenstruktur

Beim Extrahieren von Information aus Wikipedia werden die strukturiert vorliegenden Daten verwendet, die in der Wikipedia als Infoboxen, Kategorisierungen, Bilder oder

[21] http://wiki.DBpedia.org/NextSteps?v=x98 (Stand: 26.07.08)

geographische Informationen vorliegen. Diese Daten dienen der geordneten Kurzdarstellung von Artikeln und lassen sich recht unkompliziert in RDF Aussagen umwandeln. DBpedia besteht derzeit aus ungefähr 218 Millionen solcher Aussagen, die in etwa 2.18 Millionen Begriffe beschreiben.[22] Diese Begriffe besitzen eine URI, die dem relativen Pfad der URL entspricht, die zu dem gleichnamigen Wikipedia Artikel verweist. Der Wikipedia Artikel *http://en.wikipedia.org/wiki/Name* wird in DBpedia beispielsweise mit der URI *http://DBpedia.org/resource/Name* referenziert. Jeder Begriff wird desweiteren durch Eigenschaften beschrieben.

Um die Wikipedia Begriffe in eine einheitliche Klassifizierung zu bringen, werden drei unterschiedliche Klassifizierungsschemen angeboten, die wahlweise verwendet werden können [Auer, 2007]:

Wikipedia Kategorien: Wikipedia benutzt eigene Kategorien die mittels dem Simple Knowledge Organisation System Vokabular (SKOS)[23] dargestellt werden.

YAGO Klassifizierung: Die Wikipedia Kategorien wurden zudem der semantischen Datenbank YAGO[24] angepasst, die über 1.7 Millionen Entitäten besitzt.

WordNet Synset Links: [25] WordNet stellt eine lexikalische semantische Datenbank der englischen Sprache dar. Sämtliche Wikipedia Begriffe wurden dieser Datenbank angepasst.

Durch diese Klassifizierungen können Wikipedia Begriffe mit SPARQL abgefragt werden. Dies erlaubt durchaus anspruchsvolle Abfragen über Zusammenhänge, die aufgrund der Struktur der Wikipedia nicht sofort ersichtlich sind. Listing 4.5 zeigt eine YAGO Abfrage, die alle Städte ausgibt, die mehr als 2 Millionen EinwohnerInnen besitzen (sofern sie in der Wikipedia enthalten sind).

Listing 4.5: SPARQL Query in DBpedia

```
1 SELECT ?subject ?population WHERE {
2   ?subject rdf:type <http://DBpedia.org/class/yago/City108524735>.
3   ?subject DBpedia2:population ?population.
4   FILTER (xsd:integer(?population) > 2000000)
5 }
6 ORDER BY DESC(xsd:integer(?population))
```

[22] http://wiki.DBpedia.org/Datasets (Stand: 28.07.08)
[23] http://www.w3.org/2004/02/skos/ (Stand: 15.09.08
[24] http://www.mpi-inf.mpg.de/~suchanek/downloads/yago/ (Stand: 15.09.08)
[25] http://wordnet.princeton.edu/ (Stand: 28.07.08)

4.4.2 Datenzugriff

Die Datenbank erlaubt einen breit gefächerten Zugriff, da eine große Zahl an Schnittstellen zu externen technischen Lösungen besteht. Die Datensätze stehen auch als Download zur Verfügung. Um eine Seite zu erstellen, die Inhalte aus der Wikipedia extrahiert, stehen diverse SPARQL Endpoints zur Verfügung:

- Online SPARQL Endpoint[26]
- Leipzig Query Builder[27]
- iSPARQL[28]
- SNORQL[29]

Im Zuge dieses Projektes wird die Schnittstelle über das Modul SPARQL der RDF API von Drupal geknüpft.

4.5 SPARQL API

Die Drupal SPARQL API[30] basiert auf der noch in Entwicklung stehenden RDF API und ermöglicht die Abfragen nach lokal oder extern gespeicherten RDF Daten zu stellen. Die bestehende API wird im Rahmen dieser Arbeit erweitert, um als Schnittstelle für semantische Abfragen in der Datenbank DBpedia zu dienen. Im Folgenden wird das technische Konzept der API vorgestellt. Ein Überblick über die SPARQL Abfragesprache befindet sich in Abschnitt 2.3.4.

4.5.1 SPARQL API Klassen

Die API stellt eine Reihe von Klassen zur Verfügung, die zur Bildung von SPARQL Abfragen verwendet werden können. Neben dem Abfragetyp *SELECT* stehen die Abfragetypen *ASK, CONSTRUCT* und *DESCRIBE* zur Verfügung. Dazu kommen eine Reihe vom Modifikatoren um eine Suchabfrage einzugrenzen. Die Benennung entspricht

[26] http://DBpedia.org/sparql (Stand: 28.07.08)
[27] http://wikipedia.3ba.se/ (Stand: 28.07.08)
[28] http://demo.openlinksw.com/isparql/ (Stand: 28.07.08)
[29] http://DBpedia.org/snorql/ (Stand: 28.07.08)
[30] http://drupal.org/project/sparql (Stand: 28.07.08)

der SPARQL Spezifikation des W3C [Prud'hommeaux, 2008]. Die Werte der Modifikatoren werden jeweils als PHP Variablen übergeben.

4.5.2 SPARQL API Funktionen

Die Abfragen werden mittels Funktionen gebildet, die für jeden Abfragetyp existieren. Zusätzlich enthält die API Funktionen, um SPARQL Abfragen aus einem Textsegment zu parsen. Die Funktionen werden hier zusammengefasst:

- sparql_ask(): Führt eine ASK Anweisung aus.
- sparql_select($var1[, $var2, $varN, ...]): Führt eine SELECT Anweisung aus.
- sparql_construct(): Führt eine CONSTRUCT Anweisung aus.
- sparql_describe(): Führt eine DESCRIBE Anweisung aus.
- sparql_query($text[, $options, $errors]): Führt eine Anweisung in Textform aus.
- sparql_parse($text[, $options, $errors]): Parst eine Anweisung in Textform und erzeugt ein Objekt.

Listing 4.6 zeigt eine SPARQL Abfrage auf Basis dieser Funktionen, die alle Beiträge selektiert, die den Node Typ *Blog* besitzen und auf der Startseite des Systems angezeigt werden.

Listing 4.6: SPARQL API Query Beispiel

```
<?php
  // Select the titles of all blog posts promoted to the front page
  sparql_select('?title')->
    where('?node', rdf::type, rdf\_curie('drupal:node'))->
    where('?node', 'node:type', 'blog')->
    where('?node', 'node:promote', TRUE)->
    where('?node', 'node:title', rdf\_var('?title'));
?>
```

4.5.3 SPARQL API Hooks

Die API bietet lediglich eine Schnittstelle, um die Funktionalität durch externe Module zu erweitern. Der Hook *hook_sparql()* erlaubt es, Abfragen vor ihrer Ausführung zu manipulieren oder andere Aktionen bei der Ausführung zu implementieren. Der Funktion wird jeweils ein Operator und die Abfrage als Variable übergeben.

5 Praktischer Teil - Umsetzung

Dieses Kapitel umfasst die praktische Umsetzung des Moduls, das in Abschnitt 4.1 beschrieben wurde. Zunächst werden die Anforderungen an die Anwendung in einer Planungsphase definiert. Danach folgt eine Anforderungsanalyse und die Aufstellung zweier Szenarien, die dem Test des Moduls dienen. Im folgenden Entwurf wird die Programmarchitektur und ein Mock-up beschrieben. Der nächste Teil stellt die Implementierung vor, die zuletzt durch einfache BenutzerInnentests ausgewertet wird.

5.1 Planung

Wie bereits beschrieben werden die RDF und SPARQL APIs als Grundlage für ein Drupal Modul dienen, das die Schnittstelle zur Wissensdatenbank DBpedia implementiert. Über dieses Modul sollen BenutzerInnen in der Lage sein, einen neuen Inhaltstyp (siehe Abschnitt 4.2.1) zu erstellen, der es erlaubt, beliebige Daten über SPARQL Abfragen zu importieren. Hierzu muss ein neuer Node Typ (*Semantic Content*) implementiert werden. Die importierten Daten sollen in eine geeignete strukturelle Form gebracht werden, die entsprechend gewählt werden kann. Bei der Erstellung von Abfragen ist insbesondere darauf zu achten, dass die Anwendbarkeit auch ohne tiefere Vorkenntnisse möglich ist. Damit das Modul mit dem Systemkern harmoniert, muss eine Zugriffskontrolle, eine Administrationsanbindung und Installationsroutine erstellt werden.

Dieser Abschnitt besteht aus der Anforderungsdefinition, die sämtliche funktionalen und nichtfunktionalen Anwendungen des Moduls absteckt. Danach wird das Vorgehensmodell beschrieben, nach dem sich die weitere Gliederung richtet.

5.1.1 Anforderungsdefinition

Um die Aufgaben des Moduls zu spezifizieren, werden die Anwendungen in funktionale und nichtfunktionale Anforderungen getrennt:

Funktionale Anforderungen: Die funktionalen Anforderungen umfassen die Aufgaben des Moduls.

- Ein neuer Node Typ *Semantic Content* soll in das Drupal Menüsystem integriert werden.
- Durch diesen Typ sollen komplexe SPARQL Abfragen formulierbar sein, die den gesamten Datenbestand von DBpedia untersuchen.
- Die hierzu notwendige Verbindung soll durch eine Anpassung der SPARQL API erzielt werden.
- Die Bildung einer SPARQL Abfrage soll durch ein intuitives Formular unterstützt werden, das eine explorative Art der Anwendung ermöglicht.
- Eine Hilfefunktion und die Definition von Prefixes soll die Anwendbarkeit erleichtern.
- Der Node Typ soll darüberhinaus die üblichen Drupal Node Felder *Menu Settings, Input Type, Revision Information, Comment Settings, Authoring Information* und *Publishing Options* enthalten.
- Der Node Typ soll über eine Vorschaufunktion verfügen.
- SPARQL Abfragen sollen in einer zusätzlichen Datenbank abgelegt werden, um später semantisch erstellte Seiten bearbeiten zu können.
- Beim Aufruf einer durch das Modul erstellten Seite soll die gespeicherte Abfrage dynamisch ausgeführt werden, damit eine ständige Aktualisierung der Ausgabe gewährleistet ist.
- Die Datenbankmanipulation erfordert zusätzlich eine Installationsroutine des Moduls.
- Die Sichtbarkeit und Zugriffskontrolle der Module soll im Drupal Rechtesystem spezifizierbar sein.

Nichtunktionale Anforderungen: Mit den nichtfunktionalen Anforderungen werden die Eigenschaften des Moduls beschrieben.

- Das Modul soll in der Handhabung dem Drupal Kernsystem entsprechen.
- Die Handhabung soll sich an unerfahrene BenutzerInnen richten und intuitiv gestaltet sein.
- Das Modul soll auch ohne tiefgehende SPARQL Kenntnisse bedienbar sein.
- Ebenso sollen tiefgehende SPARQL Kenntnisse zu einer vielseitigeren Bedienung führen.
- Die Generierung von Inhalten soll in einer tolerierbaren Zeitspanne geschehen.
- Das Modul soll keine zusätzlichen Installationen externer Daten erfordern.
- Der Zugriff muss durch das interne Rechtesystem regelbar sein.

- Generierte Inhalte müssen bearbeitbar und verwaltbar sein.

5.1.2 Vorgehensmodell

Der Größe des Projekts entsprechend wird ein iteratives Vorgehensmodell angewandt, das sich aus einer Planungsphase, Anforderungsanalyse, Entwurf, Implementierung und Evaluation zusammensetzt. (Siehe hierzu [Boehm, 1988]) Auf diese Weise ist eine inkrementelle Verbesserung des Moduls möglich, die auf Basis der zyklischen Evaluation erfolgt. Das Vorgehensmodell wird in Abbildung 5.1 gezeigt.

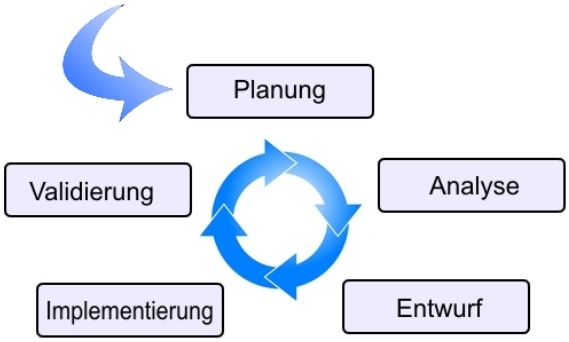

Abbildung 5.1: Vorgehensmodell auf Basis von [Boehm, 1988]

Die Strukturierung des weiteren Kapitels richtet sich ebenfalls nach den Zyklen dieses Modells. Zunächst werden konkrete Anwendungsfälle spezifiziert, die die Anforderungen an das Modul beschreiben. Danach folgt der Entwurf des technischen Grundgerüstes und die Implementierung. Zuletzt soll das Produkt anhand der Anwendungsfälle evaluiert werden. Zu diesem Zweck werden zusätzlich zwei vollständige Anwendungsszenarien kreiert, die im Rahmen des Programmtests durchlaufen werden.

5.2 Anforderungsanalyse

Die Anforderungsanalyse besteht aus der Definition von Anwendungsfällen, die die Anforderungsdefinition abdecken. Die Struktur der Anwendungsfälle gestaltet sich nach dem Schema in Tabelle 5.1. Jeder Anwendungsfall besteht aus einer Identifikationsnummer, einer Kurzbeschreibung, allfälligen Vorbedingungen, einer Beschreibung und

etwaigen Folgebedingungen. Zusätzlich werden zwei Szenarien kreiert, die beim Testen des Programms durchlaufen werden können.

Anwendungsfall	**Identifikationsnummer**
Kurzbeschreibung	Kurzbeschreibung des Anwendungsfalles.
Vorbedingungen	Vorbedingungen, damit der Anwendungsfall in Kraft treten kann.
Beschreibung	Enthält eine detaillierte Beschreibung des Anwendungsfalls.
Fehlerfall	Auswirkung im Fall eines Fehlers.
Erfolgsfall	Systemänderungen infolge einer erfolgreichen Anwendung.

Tabelle 5.1: Schema eines Anwendungsfalles

5.2.1 Anwendungsfälle

Anwendungsfall	**Inst.1**
Kurzbeschreibung	Installation des Moduls
Vorbedingungen	RDF API und SPARQL API müssen aktiviert sein.
Beschreibung	Das Modul wird im Ordner *Modules* des Installationsverzeichnisses abgelegt. Nach Aktivierung des Moduls im Administrationsmenü generiert das Installationspaket eine Datenbanktabelle um gespeicherte SPARQL Abfragen abzulegen.
Fehlerfall	Installationsfehler werden von Drupal abgefangen und ausgegeben.
Erfolgsfall	Der Node Typ *Semantic Content* ist verfügbar.

Anwendungsfall	**Gen.1**
Kurzbeschreibung	Hilfe aufrufen.
Vorbedingungen	Die Seite *Create Content/Semantic Content* wurde aufgerufen.
Beschreibung	Das ausklappbare Fenster *SPARQL Prefixes* wird angeklickt und die darin enthaltene Hilfefunktion aufgerufen. In einem neuen Fenster erscheint der Inhalt der Hilfe Datei.
Fehlerfall	-
Erfolgsfall	Hilfedatei wird angezeigt.

Anwendungsfall	**Inst.2**
Kurzbeschreibung	BenutzerInnenrechte für das Modul setzen.
Vorbedingungen	Modul ist installiert und aktiviert.
Beschreibung	Die Seite *Administer/User Management/Permissions* wird aufgerufen. Die BenutzerInnenrechte für die Erstellung von neuem Content werden gesetzt und abgespeichert. Im Frontend ist der Node Typ für ausgenommene BenutzerInnen nicht mehr sichtbar, bzw. editierbar.
Fehlerfall	-
Erfolgsfall	Das Modul kann nur von den spezifizierten Benutzergruppen aufgerufen werden.

Anwendungsfall	**Gen.2**
Kurzbeschreibung	SPARQL Abfrage über Textfeld generieren.
Vorbedingungen	Die Seite *Create Content/Semantic Content* wurde aufgerufen.
Beschreibung	Im Textfeld *SPARQL Query* wird eine gültige Abfrage verfasst. Über den Button *Preview* oder *Save* wird die Seite im Drupal System angelegt.
Fehlerfall	Leere Rückgaben oder Syntaxfehler werden im Backend ausgegeben.
Erfolgsfall	Die Seite wird im Drupal System angelegt und in der Datenbank gespeichert.

Anwendungsfall	**Gen.3**
Kurzbeschreibung	SPARQL Abfrage über Formular generieren.
Vorbedingungen	Die Seite *Create Content/Semantic Content* wurde aufgerufen.
Beschreibung	Das Formular *Query Builder* wird beschrieben und mit dem Button *Generate* abgeschickt. Im Textfeld *SPARQL Query* wird daraufhin eine gültige Abfrage generiert. Über den Button *Preview* oder *Save* wird die Seite im Drupal System angelegt.
Fehlerfall	Fehlende Eingaben werden mit Javascript abgefangen und im Frontend ausgegeben. Leere Rückgaben oder Syntaxfehler werden im Backend ausgegeben.
Erfolgsfall	Die Seite wird im Drupal System angelegt und in der Datenbank gespeichert.

Anwendungsfall	**Gen.4**
Kurzbeschreibung	Veröffentlichung einer Seite im Vorschaumodus.
Vorbedingungen	Die Seite *Create Content/Semantic Content* wurde aufgerufen und eine gültige SPARQL Abfrage generiert.
Beschreibung	Die Seite wird mit dem Button *Preview* dem System übergeben.
Fehlerfall	Der generierte Fehler wird von Drupal gemäß den Einstellungen abgefangen und im Backend angezeigt.
Erfolgsfall	Die Seite wird nach den Veröffentlichungskriterien im Backend angezeigt.

Anwendungsfall	**Gen.5**
Kurzbeschreibung	Seite veröffentlichen.
Vorbedingungen	Die Seite *Create Content/Semantic Content* wurde aufgerufen und eine gültige SPARQL Abfrage generiert.
Beschreibung	Die Seite wird mit dem Button *Save* dem System übergeben. Die SPARQL Eingaben werden in der Datenbank gespeichert.
Fehlerfall	Der generierte Fehler wird von Drupal gemäß den Einstellungen abgefangen und im Backend angezeigt.
Erfolgsfall	Die Seite wird nach den Veröffentlichungskriterien im Frontend angezeigt.

Anwendungsfall	**View.1**
Kurzbeschreibung	Veröffentlichte Seite aufrufen.
Vorbedingungen	Eine gültige Seite wurde mit dem *Semantic Content Modul* angelegt.
Beschreibung	Eine über das Modul erstellte Seite wird aufgerufen. Die View Funktion der Implementierung führt die in der Datenbank gespeicherte SPARQL Abfrage dynamisch aus und generiert aus der Rückgabe die angezeigte Seite.
Fehlerfall	Leere Rückgaben oder Syntaxfehler werden im Backend ausgegeben.
Erfolgsfall	Die Seite wird im Body Teil des Frontend angezeigt.

Anwendungsfall	**View.2**
Kurzbeschreibung	Veröffentlichte Seite bearbeiten.
Vorbedingungen	Eine gültige Seite wurde mit dem *Semantic Content Modul* angelegt und aufgerufen.
Beschreibung	Die Seite wird mit dem Button *Edit* angewählt. Daraufhin wird die SPARQL Abfrage aus der Datenbank gelesen und im Textfeld angezeigt. Ebenso werden die Inhalte aus der Drupal Datenbank ausgelesen und in die entsprechenden Felder geladen.
Fehlerfall	Fehler bei der Datenbankverbindung werden im Textfeld ausgegeben.
Erfolgsfall	Die Seite wird im Bearbeitungsmodus angezeigt.

5.2.2 Szenarien

Die folgenden Szenarien werden als Basis für die Programmtests des Plug-ins herangezogen und sollen im Anschluss durchlaufen werden. Daher wird darauf geachtet, eine möglichst große Zahl an Anwendungen abzudecken.

Szenario 1: Installation und einfache Anwendung

Testbenutzerin A möchte das Modul aus dem Drupal Modularchiv herunterladen und eine einfache semantische Seite erstellen. Die Drupal Grundinstallation ist bereits lauffähig und die RDF und SPARQL APIs installiert. Sie folgt dem Link auf der Drupal Webseite und betätigt den Download Link. Sie öffnet das heruntergeladene Zip Archiv und liest die Readme.txt Datei. Daraufhin speichert sie das heruntergeladene Verzeichnis im Ordner *Modules* der Drupal Grundinstallation. Sie öffnet den Administrationsbereich und sieht das verfügbare Modul in der Liste aller Module. Sie aktiviert das entsprechende Kontrollkästchen und speichert die Einstellungen woraufhin das Modul installiert wird.

Damit ist nun ein neuer Node Typ *Semantic Content* verfügbar, über den die Benutzerin eine Testseite erstellen möchte. Da die Webseite über asiatische Länder handeln soll, möchte die Benutzerin eine Liste aller ostasiatischen Länder importieren. Sie gibt einen Titel ein und öffnet das ausklappbare Fenster *SPARQL Prefixes*. Daraufhin besucht sie die URL *http://DBpedia.org/resource/Japan* um die verfügbaren Eigenschaften der Resource *Japan* einzusehen. Sie befüllt die Felder des Query Builders wie in Abbildung 5.2 dargestellt.

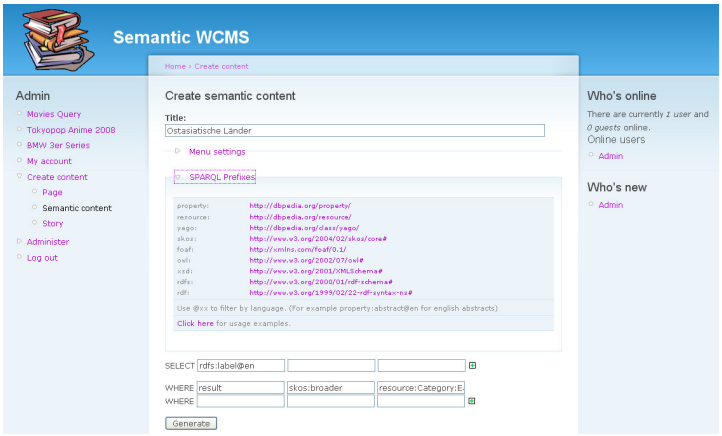

Abbildung 5.2: Semantic Content Seite mit ausgeklappten Prefixes.

Nun betätigt Benutzerin A den *Generate* Button woraufhin das System die SPARQL Abfrage aus Listing 5.1 generiert. Im Vorschaumodus erhält sie das Ergebnis ihrer Abfrage und setzt die Seite in den Menüeinstellungen auf die oberste Position. Nach dem Abspeichern der Seite werden die Daten im Drupal System angelegt und ein Link zu der erstellten Seite erscheint in der Menüleiste. Im Frontend ist nun die erstellte Seite mit der Liste aller ostasiatischer Länder, wie in Abbildung 5.3 zu sehen.

Listing 5.1: SPARQL Abfrage aus Szenario 1

```
SELECT ?label
WHERE{
  ?result <http://www.w3.org/2004/02/skos/core#broader>
  <http://DBpedia.org/resource/Category:East_Asian_countries>
 OPTIONAL{
  ?result <http://www.w3.org/2000/01/rdf-schema#label> ?label FILTER (LANG(?label) = "
       en")
 }
}
```

Szenario 2: Erstellung und Verwaltung komplexer Seiten

Testbenutzer B besitzt bereits eine Drupal Seite mit allen nötigen Installationen. Da Benutzer B großes Interesse für Filme von Steven Spielberg besitzt, möchte er eine Seite erstellen, die alle Titel und Beschreibungen der Filme des Regisseurs enthält. Daher

Abbildung 5.3: Erstellte Seite aus Szenario 1.

öffnet er eine neue Seite des Typs *semantic content* und gibt die Formulardaten aus Listing 5.2 in den Query Builder ein.

Listing 5.2: SPARQL Abfrage aus Szenario 2

```
1 SELECT ?property:title@en ?property:abstract@en
2 WHERE ?movie property:director resource:Steven_Spielberg
```

Durch einen Klick auf den Button *Generate* generiert das System die entsprechende SPARQL Abfrage und speichert sämtliche Daten in der temporären Datenbank zur späteren Verfügbarkeit ab. Da Benutzer B auch gerne Feedback für seine Seite bekommen möchte, lässt er Kommentare für seine Seite zu. Er füllt nun die Kommentaroptionen aus und platziert seine Seite im Menüsystem von Drupal. Nach erfolgter Speicherung erscheint ein Link zu seiner Seite in der Navigationsleiste. Benutzer B entschließt sich nun die Seite zu editieren, und betätigt das Registerfeld *edit*. Danach wird die Seite mit sämtlichen Formularfeldern erneut geladen. Der Benutzer erweitert seine Eingaben wie in Listing 5.3 beschrieben und erhält die darunter stehende SPARQL Abfrage.

Listing 5.3: Verbesserte SPARQL Abfrage aus Szenario 2

```
1 SELECT ?property:title@en ?property:abstract@en ?property:budget
2 WHERE ?movie property:director resource:Steven_Spielberg
3 WHERE ?movie property:music resource:John_Williams
4 WHERE ?movie property:distributor resource:Universal_Pictures
5
6 SELECT ?title ?abstract ?budget
7 WHERE{
```

```
8   ?movie <http://DBpedia.org/property/director> <http://DBpedia.org/resource/
        Steven_Spielberg>;
9   <http://DBpedia.org/property/music> <http://DBpedia.org/resource/John_Williams>;
10  <http://DBpedia.org/property/distributor> <http://DBpedia.org/resource/
        Universal_Pictures>
11  OPTIONAL{
12    ?movie <http://DBpedia.org/property/title> ?title FILTER (LANG(?title) = "en")
13    ?movie <http://DBpedia.org/property/abstract> ?abstract FILTER (LANG(?abstract) = "
          en")
14    ?movie <http://DBpedia.org/property/budget> ?budget
15  }
16 }
```

Abbildung 5.4 zeigt die ausgefüllte Seite zu diesem Zeitpunkt. Benutzer B geht in den Vorschaumodus um die Ergebnisse zu betrachten. Sofern er nicht mit dem Ergebnis zufrieden sein sollte kann durch einfache Änderungen im Formular mit der Ausgabe experimentiert werden. Er speichert die Seite erneut ab und ruft die Seite über ihren Link im Navigationsbereich auf. Die erstellte Seite enthält nun alle Titel, Beschreibungen und Budgetdaten der Filme von Steven Spielberg mit dem Soundtrack von John Williams von Universal Pictures.

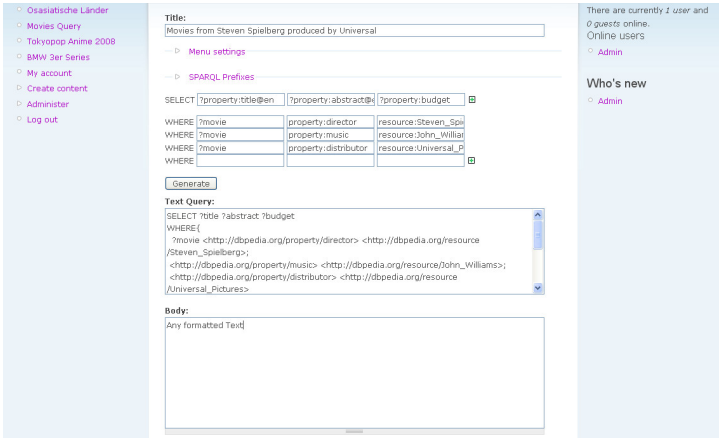

Abbildung 5.4: Verbesserte Abfrage aus Szenario 2.

Für andere BenutzerInnen, die die Seite zu einem späteren Zeitpunkt aufrufen, bleibt der Inhalt ständig aktuell, da er bei jedem Aufruf dynamisch aus DBpedia generiert wird. Jede Änderung in DBpedia wird somit automatisch übernommen. Ein weiterer Benutzer besucht die Seite und möchte ein Kommentar abgeben. Also betätigt er den

Link am Ende der Seite und speichert seine Meinung ab. Die generierte Seite mit einem exemplarischen Kommentar wird in Abbildung 5.5 gezeigt.

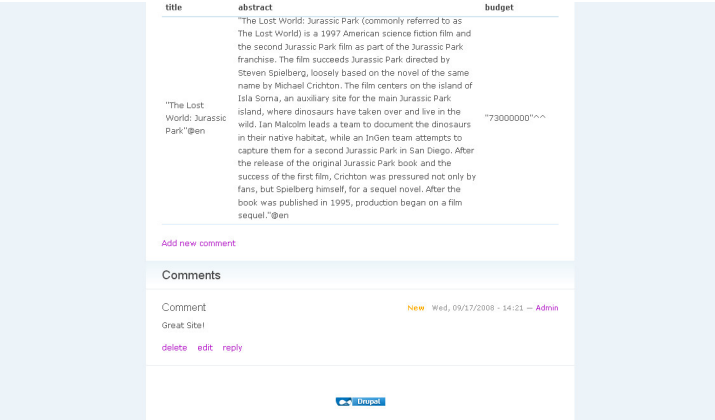

Abbildung 5.5: Ansicht der Ergebnisseite mit Kommentar.

5.3 Entwurf

Der Entwurf des Moduls umfasst die Beschreibung der Programmarchitektur, und der Dateistruktur des Moduls. Ebenso werden die zu implementierenden Drupal Hooks beschrieben, über die die Kommunikation mit dem Drupal System erfolgen wird. Danach wird auf das Konzept der Datenbank eingegangen, die als Speicher für die Formulardaten und SPARQL Abfragen dienen wird. Zuletzt erfolgt die Erstellung eines Mock-ups, der als Grundlage für die Implementierung des User Interface herangezogen werden soll.

5.3.1 Systemdesign

Das Modul besteht aus einer *.modul* Datei, die als Verbindung zum Drupal Kern fungiert und die Mehrzahl der nötigen Hooks enthält. Daneben ist eine Installationsroutine, Installationshinweise, eine Hilfefunktion, und eine Anbindung an die SPAQRL API erforderlich. Das Modul enthält neben diesen nötigen Drupal Komponenten ein Ajax Script, das über eine Javascript mit der *.modul* Datei kommuniziert. Auf diese Weise können SPARQL Abfragen über ein Formular asynchron generiert werden. Die Umsetzung der definierten Anforderungen kann über unterschiedliche Hooks erreicht werden. Abbildung

5.6 zeigt die Architektur des Systems und die Verbindungen zum Drupal CMS, der internen Datenbank und der semantischen Datenbank DBpedia. Eine genauere Beschreibung der einzelnen Funktionen des Moduls erfolgt in Abschnitt 5.4.

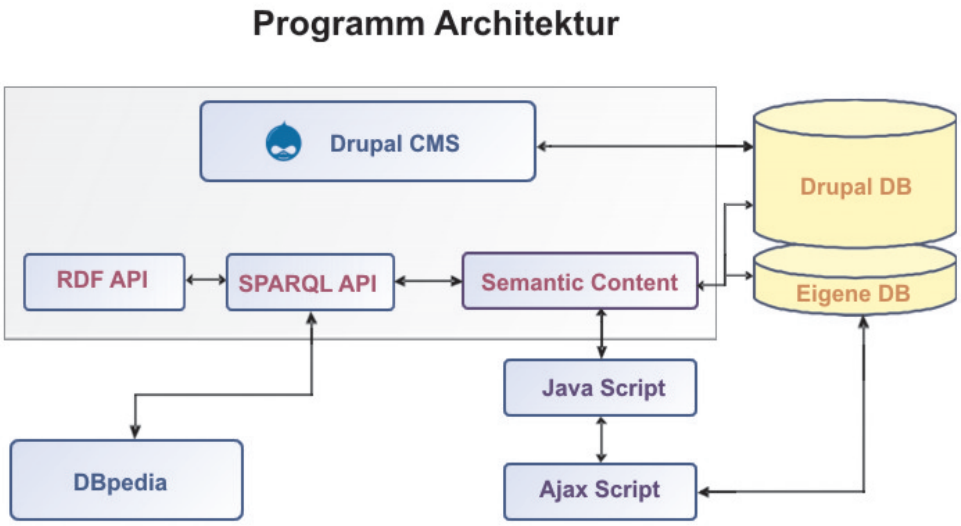

Abbildung 5.6: Programmarchitektur und Positionierung im Drupal System

Die Dateistruktur des Moduls wird in Abbildung 5.7 dargestellt. Die Namensgebung der Dateien wird vom Drupal System vorgegeben, damit das Modul erkannt und geladen werden kann. Daher tragen alle von Drupal angesteuerten Dateien den Modulnamen *semantic_content*. Lediglich externe Komponenten wie das Ajax-PHP Script können individuell bezeichnet werden. Die einzelnen Komponenten werden im Folgenden kurz beschrieben:

semantic_content.module: Die Hauptdatei des Moduls enthält die gesamte Kommunikation mit dem Drupal Kern und implementiert die erforderlichen Hooks. Dazu gehört in erster Linie die Implementierung des neuen Node Typs und des zugehörigen Formulars, die Kommunikation mit der Datenbank, die Anbindung an die SPARQL API und die Verwaltung der Zugangsrechte im System. Die folgende Liste enthält alle vorgesehenen Funktionen:

- hook_node_info: Definition des Node Typs und Einbindung externer Scripts.
- hook_theme: Registrierung eines Ausgabeschemas.
- hook_perm: Definition der BenutzerInnenrechte.
- hook_access: Implementierung der BenutzerInnenrechte.

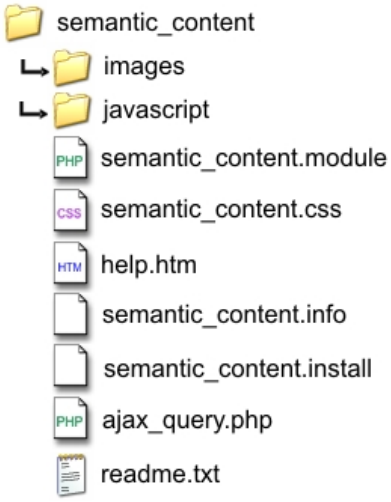

Abbildung 5.7: Dateistruktur des Moduls

- hook_form: Implementierung des Formulars und Registrierung der Komponenten.
- hook_load: Ladefunktion externer Daten.
- hook_insert: Speicherung zusätzlicher Daten in der Datenbank.
- hook_update: Aktualisierung zusätzlicher Daten.
- hook_delete: Löschvorgang zusätzlicher Daten
- hook_view: View Funktion die aus der Datenbank gelesene SPARQL Abfragen dynamisch ausführt.
- hook_help: Hilfefunktion.
- node_result: Führt eine Abfrage über die SPARQL API aus und gibt ein Array an die Theme Funktion zurück.

semantic_content.css: Das Cascade Stylesheet definiert das Layout der Seite. Somit können einzelne Formularelemente ein individuelles Format erhalten, das nicht von der globalen CSS Datei abhängig ist. Dies ist notwendig um eine freie Positionierung und Absatzgestaltung für das Formular festzulegen.

help.htm: Die Hilfedatei enthält Hinweise zur Handhabung des Moduls. Sie kann über den neuen Node Typ aufgerufen werden und wird dann in einem neuen Fenster angezeigt.

semantic_content.info: Die Infodatei wird von Drupal benötigt, um das Modul zu

erkennen. Sie enthält Versionshinweise und eine Beschreibung des Moduls für den Administrationsbereich.

semantic_content.install: Die Installationsroutine legt eine neue Datenbanktabelle im System an. Ebenso werden Update- und Löschfunktionen definiert. Die Datei wird erstmalig bei der Aktivierung des Moduls aufgerufen.

ajax_query.php: Das serverseitige Ajax Script dient der Manipulation des Formulars. Hierin wird ein SPARQL Query aus den übermittelten Formulardaten erstellt und ein neuer Datensatz in der Datenbank angelegt. Somit kann bereits im Vorschaumodus auf die Datenbank zugegriffen werden, ohne dass die Seite im Drupal System abgespeichert werden muss.

ajax.js: Das Java Script ist für die Kommunikation mit Ajax erforderlich. Es definiert das XHTTP Request Object[1] und leitet die Rückgabe des Ajax PHP Scripts an das Formular.

readme.txt: Die Readme Datei enthält Hilfe zur Installation des Moduls.

5.3.2 Datenbank

Die Einrichtung zusätzlicher Datenbanktabellen wird notwendig, um SPARQL Abfragen dynamisch generieren und abspeichern zu können. Die Drupal Datenbank stellt hierzu nicht ausreichende Funktionalität zur Verfügung. Auch das Formular zur Generierung von SPARQL Abfragen kann nicht über Drupal Standardformularfelder erfolgen, da das Drupal System keine externen Ajax Scripts unterstützt. Der Zugriff über Java Script muss daher über einen Kunstgriff erfolgen, indem gewöhnliche HTML Formularfelder als Suffix eines Drupal Formularfelds in das Skript eingebunden werden. Diese Formularfelder müssen ebenfalls in der Datenbank berücksichtigt werden. Die Datenbankstruktur wird in Tabelle 5.2 beschrieben.

5.3.3 Mock-up

Zur Veranschaulichung des Interfaces wurde ein Mock-up erstellt, der den Aufbau des Formulars zeigt, wenn der Node Typ *Semantic Content* aufgerufen wird. Abbildung 5.8 zeigt die Raumaufteilung des Node Typs in einer Standard Drupal Umgebung. Neben

[1] http://www.w3.org/TR/XMLHttpRequest/ (Stand: 15.09.08)

Tabelle 1:

Tabellenfeld	Typ	Beschreibung
session_key	VARCHAR(20)	Identifikator
query	LONGTEXT	SPARQL Query
body	LONGTEXT	Body Teil der Seite
selects	INTEGER	Anzahl der select Felder
wheres	INTEGER	Anzahl der where Felder

Tabelle 2:

Tabellenfeld	Typ	Beschreibung
session_key	VARCHAR(20)	Identifikator
field	VARCHAR(90)	Inhalt des Feldes
type	VARCHAR(20)	Typ des Feldes

Tabelle 5.2: Datenbankschema der Semantic Content Datenbank

dem Titel und Body Teil, enthält das Interface einen Bereich, in dem SPARQL Abfragen als RDF Triples beschrieben werden können. Die verwendbaren Prefixes werden in einem zusätzlichen Fenster angezeigt. Nachdem der *Generate* Button betätigt wurde, soll die SPARQL Abfrage dynamisch in das Textfeld *SPARQL Query* geladen werden. Hierzu werden die Daten vom Ajax Script ausgewertet. Der Body Teil kann zur Formatierung der Seite genutzt werden.

Der Mock-up des Query Builders ist in Abbildung 5.9 zu sehen. Er enthält jeweils Gruppen von drei Textfeldern, mit denen RDF Triples gebildet werden können. Mit einem Additionssymbol können neue Gruppen in das Formularfeld geladen werden. Die nötige Manipulation des Formulars wird ebenfalls über das Ajax Script erreicht. Eine korrekt gebildete SPARQL Abfrage wird in das darunterliegende Textfeld geladen, wo sie adaptiert werden kann. Sollte die Vorschaufunktion benutzt werden, müssen alle Formularfelder geladen werden damit die Abfrage weiterhin bearbeitbar bleibt. Da das Drupal System Daten jedoch vor dem Abspeichern nur im internen Zwischenspeicher ablegt, müssen sämtliche Felder bereits nach der Eingabe mittels Ajax in den moduleigenen Datenbanktabellen abgespeichert werden.

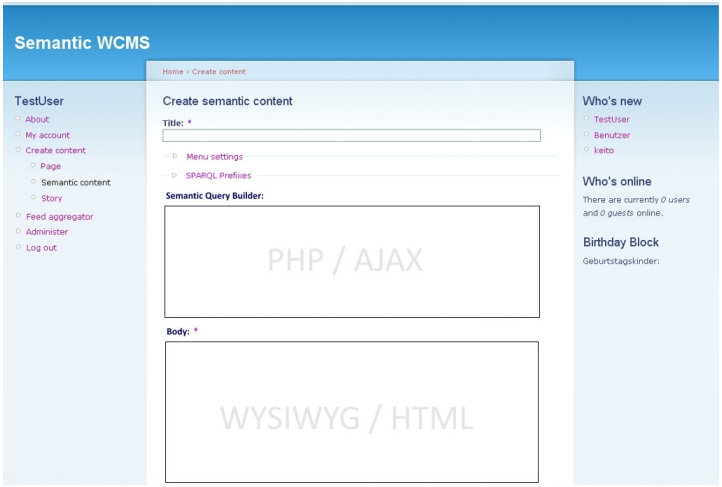

Abbildung 5.8: Mock-up der Gesamtansicht

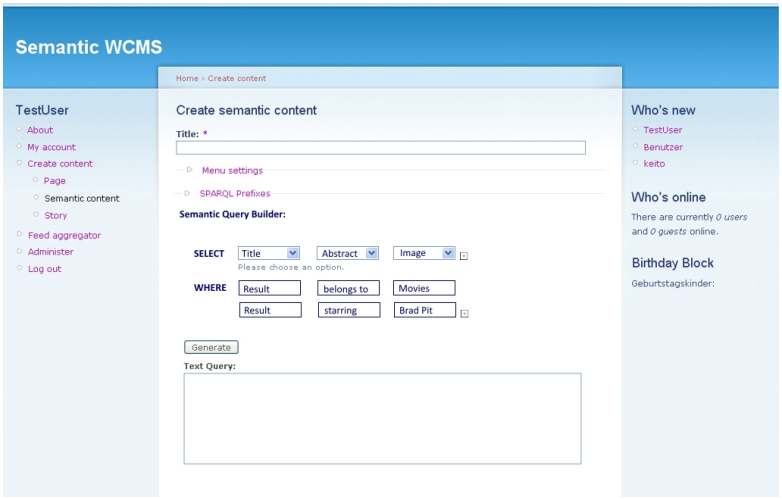

Abbildung 5.9: Mock-up des Query Builders

5.4 Implementierung

Im nun folgenden Abschnitt wird die Realisierung des vorgestellten Konzeptes aus technischer Sicht beschrieben. So soll ein Einblick in die Entwicklung des Moduls gewonnen werden. Für eine Dokumentation aus AnwenderInnensicht sei auf die frei verfügbare Online Version der Implementierung verwiesen (siehe Abschnitt 5.4.1), da eine solche Beschreibung den Rahmen dieser Arbeit sprengen würde. Die Strukturierung des Kapitels richtet sich nach dem Benutzungsablauf des Moduls und beginnt bei der Installation, umfasst die technischen Zusammenhänge bei der Erstellung einer semantischen Seite und beschreibt letztlich das Datenmanagement, das zum Speichern und Editieren der Daten notwendig ist.

5.4.1 Installation

Die Installation erfolgt einfach durch das Ablegen des Moduls im Ordner *modules* der Drupal Grundinstallation. Das System sucht dann nach der *.info Datei, die das Modul im System registriert. Die *.info Datei folgt dem in Abschnitt 4.2.3 beschriebenen Aufbau. Somit ist das Modul im Backend des CMS aktivierbar. Geschieht dies, wird nach der *.install Datei des Moduls gesucht und die darin enthaltene Funktion *hook_install()* ausgeführt. Diese führt eine Datenbankabfrage aus, die die Datenbanktabellen (siehe 5.2 generiert. Bei einer Löschung des Moduls wird entsprechend die *hook_uninstall()* Funktion aufgerufen, die alle zusätzlichen Tabellen entfernt. Das nun aktivierte Modul wird von Drupal ausgeführt, indem alle Hooks der *.module Datei über Listener überwacht werden.

Verwaltung der Zugangsrechte

Als nächstes muss die Verwaltung der Zugangsrechte im System behandelt werden. Die PHP Funktion *hook_perm()* definiert vier Rechtegruppen, die in Listing 5.4 dargestellt werden.

Listing 5.4: semantic_content_perm()

```
1 <?php
2   function semantic_content_perm() {
3     return array('view semantic content', 'create semantic content',
4                  'edit own semantic content', 'delete semantic content'); }
5 ?>
```

Nun können die Rechte im Administrationsbereich zwischen registrierten und anonymen BenutzerInnen aufgeteilt werden. Damit die Rechteverteilung vom System übernommen wird, müssen die erstellten Gruppen jeweils einer Drupal Operation zugeordnet werden. Dies wird in Listing 5.5 gezeigt. Die Funktion fragt über die globale Variable *$user* ab, welche Rechte für BenutzerInnen bestehen. Die Rückgabe gibt dem System an, wie die Operation behandelt werden soll. Das System kennt hierbei die vier Operationen *view, create, delete* und *update*.

Listing 5.5: semantic_content_access()

```
<?php
  function semantic_content_access($op, $node) {
    global $user;
    // Beispiel einer Operationszuordnung
    if($op == 'view'){
      return user_access('view semantic content');}
    }
  }
?>
```

Testinstallation

Das gesamte System mit installierten Komponenten ist im Internet frei zum Test unter der Adresse http://semantic.site40.net verfügbar[2]. Außerdem kann das Paket heruntergeladen werden[3]. Die Nutzung setzt eine lauffähige Installation des Drupal Systems 6.x[4] voraus und erfordert die ebenfalls frei erhältlichen Module der RDF API[5] und SPARQL API[6]. Weitere Installationshinweise sind im Downloadarchiv enthalten.

5.4.2 Erstellung des Node Typs

Der Node Typ *Semantic Content*, der die Verbindung zur SPARQL API herstellt, wird vom Hook *hook_node_info()* initialisiert, von dem aus alle übrigen Skripts eingebunden werden. Der Aufbau des Hooks wird in Listing 5.6 dargestellt. Der Import der SPARQL

[2] http://semantic.site40.net (Stand: 15.09.08)
[3] http://semantic.site40.net/downloads/semantic_content.rar (Stand: 15.09.08)
[4] http://drupal.org/drupal-6.2 (Stand: 15.09.08)
[5] http://drupal.org/project/rdf (Stand: 15.09.08)
[6] http://drupal.org/project/sparql (Stand: 15.09.08)

API macht die dort definierten Funktionen für das gesamte Modul verfügbar. Die CSS Datei wird bei der Erstellung des Formulars für den Inhaltstyp benötigt. Das folgende Array gibt dem System an, welche Standardkomponenten im neuen Node Typ von Drupal übernommen werden. In diesem Fall wird der Titel und Body Teil des Standardformulars übernommen. Die weiteren Einstellungen betreffen die Darstellung des Node Typs auf der *Create Content* Seite des CMS.

Listing 5.6: semantic_content_node_info()

```
<?php
  function semantic_content_node_info(){
    require_once drupal_get_path('module', 'sparql') . '/sparql.inc';
    // Node Typ Standard Elemente definieren
    return array(
      'semantic_content' => array(
      'name' => 'semantic content',
      'module' => 'semantic_content',
      'has_title' => TRUE,
      'has_body' => TRUE,)); }
?>
```

Formular

Der Inhalt des Node Typs wird im Hook *hook_form()* implementiert. Hier werden alle Formularfelder angelegt und zusätzliche Operationen ausgeführt. Listing 5.7 enthält das Schema dieser Implementierung. Zunächst wird ein Sitzungsschlüssel generiert, der als Identifikator für die Kommunikation mit einem externen Ajax Skript dient. Alle temporär gespeicherten Daten erhalten diesen Schlüssel. Nun werden alle bereits angelegten Daten aus der Datenbank ausgelesen und in die Formularfelder geladen, damit auch in der Vorschaufunktion mit externen Formularfeldern gearbeitet werden kann. Im zweiten Teil des Hooks werden Drupal-interne Formularelemente definiert, die in der internen Datenbank abgelegt werden können. Das Formularelement *query_builder* enthält den SPARQL Query Builder, über den SPARQL Abfragen generiert werden können.

Listing 5.7: semantic_content_form()

```
<?php
  function semantic_content_form(&$node) {
    $type = node_get_types('type', $node);

    // Sitzungsschlüssel generieren
    $node->session_key = $session_key;
```

```
7     // Zusätzliche Daten aus der Datenbank auslesen
8     $query = db_fetch_array(db_query('query'));
9     // Daten zuweisen
10    if(!isset ($node->textarea)){
11      $node->textarea = $query['body'];
12    }
13    // Drupal Formularfelder definieren
14    $form['title'] = array(
15      '#type'=> 'textfield',
16      '#default_value' => $node -> title,);
17    $form['query_builder'] = array(
18      '#title' => t('SPARQL Prefixes'),
19      '#type' => 'fieldset',
20      '#description' => t($sparql_prefixes),
21      '#suffix' => $fields,); //Zusätzliche Formularelemente integrieren
22    $form['session_key'] = array(
23      '#type' => 'hidden',
24      '#default_value' => $node->session_key,);
25    return \$form;
26  }
27 ?>
```

Das vollständig implementierte Formular ist in Abbildung 5.10 zu sehen. Es besteht aus einem Titel, einem ausklappbaren Formularfeld, das die SPARQL Prefixes beschreibt, dem Query Builder und einem Textfeld für den Body Teil des Dokuments. Die Standard Drupal Komponenten, werden aus der Klasse Node abgeleitet und werden direkt vom CMS verarbeitet. Das Modul muss sich jedoch um die korrekte Speicherung der Formulardaten, und die Anzeige kümmern, die vom Formular generiert werden soll.

Semantic Query Builder

Der Semantic Query Builder ist jener Bereich des Formulars, in dem SPARQL Abfragen aus RDF Triples gebildet werden. Um das zu erreichen, wird ein externes Ajax Skript eingebunden, das aus einem PHP Skript und einer Javascript Datei besteht, die für die Verbindung zwischen diesen Teilen sorgt. Die Javascript Datei greift über die ID Bezeichnung der Formular Felder auf deren Inhalt zu und schickt diese an eine PHP Datei, die jene Daten in eine textbasierte SPARQL Abfrage umsetzt. Dazu ist lediglich eine Parsing Funktion notwendig, die die erlaubten Prefixes in ausgeschriebene Adressen übersetzt und die Datensätze strukturiert.

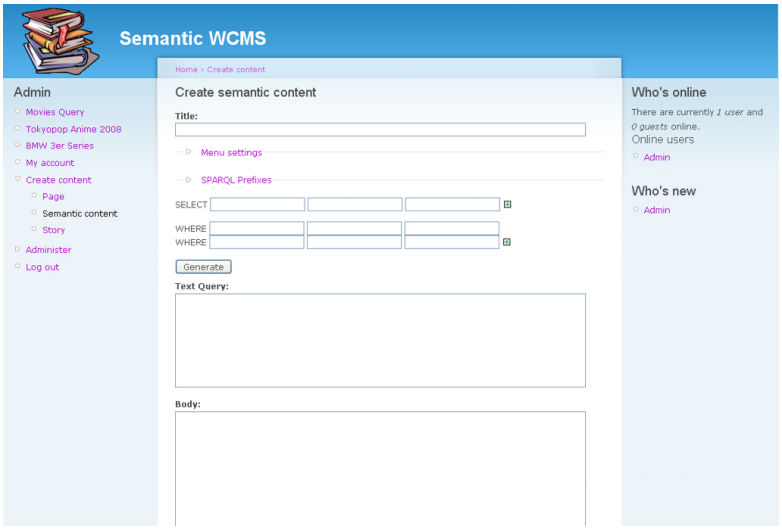

Abbildung 5.10: Ansicht des implementierten Formulars

Ebenso sorgt das Ajax Skript dafür, dass die Teftfeldgruppen des Query Builder erweitert werden können. Hierzu wird ein versteckter HTML Tag des Typs ** im Formular durch zusätzliche Formularfelder ersetzt. Die asynchrone Einbindung wird auch hier über Javascript erreicht. Sobald die SPARQL Abfrage in textueller Form vorliegt, kann sie in der Datenbank abgelegt werden. Das Kommunikationsschema zwischen den Komponenten wird durch Abbildung 5.11 schematisiert.

Sobald das Formular abgeschickt wird, greift der Drupal Listener auf die Implementierung der Funktion *hook_insert()* zu, die die Daten in der Datenbank ablegt. Wird jedoch nur der Vorschaumodus aktiviert, verwendet das Drupal System einen Zwischenspeicher um die Inhalte darzustellen. Die in der Node definierten Formularfelder können dem Zwischenspeicher nicht zugewiesen werden, daher werden alle Formularfelder bereits beim Erstellen einer SPARQL Abfrage mit dem Query Builder in der Datenbank abgelegt. Durch die Ladefunktion im Hook *hook_form()* kann auf diese Daten zugegriffen werden wenn sich das System im Vorschaumodus befindet. Wird die Seite tatsächlich angelegt, werden die Felder in der internen Drupal Datenbank gespeichert.

Beim Aufruf des Bearbeitungsmodus einer Seite durch den Button *edit*, müssen, sofern die zugeteilten BenutzerInnenrechte es gestatten, alle Daten durch die Implementierung der hooks *hook_load* und *hook_update* erneut berabeitet werden.

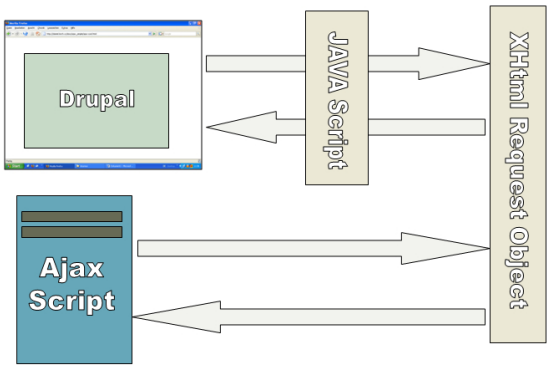

Abbildung 5.11: Kommunikationsschema zwischen der Modul Datei und externem Skript.

Ausführen der SPARQL Abfrage

Die Auswertung der SPARQL Abfrage erfolgt erst in der View Funktion, da somit gewährleistet ist, dass die Abfrage bei jedem Aufruf der Node neu ausgeführt wird. An dieser Stelle wird auch auf die SPARQL API zugegriffen, die die übermittelten Daten in eine interne Datenstruktur überführt, die durch die RDF API festgelegt wird. Listing 5.8 enthält den Aufbau dieser Funktion.

Listing 5.8: semantic_content_view()

```
<?php
  function semantic_content_view($node, $teaser = FALSE, $page = FALSE) {
    $node = node_prepare($node, $teaser);
    // SPARQL Abfrage ausführen
    $node->content['sparql_results'] = array(
      '#value' => theme('sparql_results', semantic_content_node_result($node)), );
    // Zusätzliche Felder
    $node->content['textarea'] = array(
      '#value' => $node->textarea, );
    return \$node;
  }
?>
```

Da alle Daten als Array in einer Node gespeichert werden, müssen sie für eine Ausgabe dem Feld *content* dieses Arrays zugewiesen werden. Die Art der Formatierung kann durch den Hook *hook_theme()* geändert werden, was in diesem Fall für die Rückgabe

der SPARQL Ergebnisse geschieht. Dort wird das Rückgabe Array in eine tabellarische Form umgewandelt.

In der Funktion *semantic_content_node_result()* werden die durch die SPARQL API aufbereiteten RDF Daten an den öffentlichen Endpunkt von DBpedia geschickt und unformatiert zurückgegeben. Sollte es zu einem Fehler oder einer leeren Rückgabe kommen, wird der Fehler an dieser Stelle abgefangen und ausgegeben. Im Fall einer ungültigen Abfrage gibt das System *No Success* aus, bei einer leeren Rückgabe wird *No Results* angezeigt.

5.5 Programmtests

Im letzten Abschnitt des praktischen Teils soll das Endprodukt einem informellen Programmtest unterzogen werden, der sich aus den in Abschnitt 5.2.2 erstellten Szenarien ableitet. Zunächst werden das Setting dieses Tests vorgestellt und die entsprechenden Aufgaben definiert. Im Anschluss wird das Ergebnis besprochen. Um einen kritischen Blick aus AnwenderInnensicht zu erhalten, wurde das Modul im Entwicklungsforum von Drupal[7] vorgestellt und versucht konstruktive Meinungen einzuholen, die danach kurz beleuchtet werden sollen. Abschließend wird das Ergebnis des praktischen Teils zusammengefasst.

5.5.1 Setting

Das Setting des Programmtests besteht aus einer Aufgabenstellung und einer Beschreibung der Testumgebung. Ebenso werden die testenden Personen vorgestellt, um deren Vorkenntnisse und die Voraussetzungen heraus zu streichen.

Aufgabenstellung

Die Aufgabenstellung lässt sich in drei Teilaufgaben gliedern. Diese umfassen die Erstellung einer semantischen Seite durch eine SPARQL Abfrage, die Bearbeitung einer vorhandenen Seite und die Möglichkeit zu einer freien Gestaltung von semantischen Abfragen. Die Aufgaben werden nun im Detail vorgestellt:

[7] http://drupal.org/forum (Stand: 01.10.08)

Aufgabe 1: Einfache Seite erstellen In der ersten Aufgabe soll eine einfache semantische Seite mit einer gültigen Rückgabe erstellt werden.

- Eine neue semantische Seite soll über *Create Content/Semantic Content* aufgerufen werden.
- Das ausklappbare Fenster *SPARQL Prefixes* soll geöffnet und die Hilfedatei aufgerufen werden.
- Nun soll das Formularfeld mit der Abfrage aus Listing 5.2 befüllt werden um eine Seite aller Filme des Regisseurs Steven Spielberg zu generieren.
- Mit dem *Generate* Button soll nun aus dem Formular eine Abfrage generiert werden.
- In den Menüeinstellungen soll die neue Seite mit einem beliebigen Titel an die oberste Stelle gesetzt werden. Durch Drücken des Save Buttons wird die Seite im System angelegt.
- Die angelegte Seite soll nun über das Menüsystem aufgerufen werden.

Aufgabe 2: Seite editieren In der zweiten Aufgabenstellung soll die erstellte Seite aufgerufen, kommentiert und editiert werden, um die Ausgabe zu erweitern.

- Auf der aufgerufenen Seite soll nun ein Kommentar abgegeben werden.
- Als nächstes soll das Registerfeld *edit* angewählt werden.
- Das geladene Formular soll durch eine zusätzliche Zeile erweitert werden. Hierzu kann Listing 5.3 herangezogen werden.
- Die überarbeitet Seite soll nun im Vorschaumodus betrachtet und anschließend veröffentlicht werden.

Aufgabe 3: Freie Benutzung Als dritte Aufgabe soll zuletzt eine beliebige Seite erstellt werden. Diese Seite soll ebenfalls im Menüsystem von Drupal angelegt werden. Als Hilfestellung kann die Hilfedatei verwendet werden. Die erstellte SPARQL Abfrage soll zumindest zwei SELECT Felder und eine WHERE Gruppe enthalten.

Vorbereitung

Um die Aufgabe durchzuführen, wurde eine vorinstallierte Testversion des Systems mit den lauffähigen APIs zur Verfügung gestellt. Die Testpersonen erhielten einen Ausdruck der Aufgaben die ohne Zeitbegrenzung durchlaufen werden konnten. Während der Aufgaben konnte kommuniziert werden, um Fragen zu stellen, oder das Vorgehen zu kommentieren. Auf diese Weise sollte ein informelles Feedback zum Programm und der Art der

Anwendung gewonnen werden. Da die Vorkenntnisse und Voraussetzungen der Personen bei diesem Test eine Rolle spielen, werden die testenden Personen nun kurz vorgestellt.

Handelsangestellte ohne CMS Vorkenntnisse: Gabriela ist im Einzelhandel beschäftigt und eine regelmäßige Internet Nutzerin. Sie ist eine Testanwenderin mit geringen Erfahrungen im Bereich des Content Management, benutzt das Internet primär als Anwenderin und verfügt über mehrere Konten bei Internet Gemeinschaften. Bei der Benutzung des Semantic Content Moduls ist sie auf die Unterstützung der verfügbaren Hilferessourcen angewiesen. Sie möchte das Modul nutzen, um eine möglichst einfache Seite zu einem ihrer Interessensgebiete zu erstellen.

Informatikstudent mit CMS Erfahrungen: Peter ist ein Informatikstudent mit guten Kenntnissen in Content Management Systemen. Da er eine eigene Internet Seite betreibt, ist er auch mit der Verwaltung und Entwicklung von CMS vertraut. Die Installation und Aktivierung eines Moduls stellt für ihn kein Problem dar. Peter verfügt jedoch über keine Erfahrungen mit SPARQL und möchte daher ebenso in erster Linie eine lauffähige Seite generieren, um seine Kenntnisse zu vertiefen.

5.5.2 Ergebnisse

Die Tests mit den ProbandInnen brachten einige interessante Ergebnisse. Es zeigte sich, dass der Hilfestellung durch die verfügbaren Hilferessourcen bei beiden Test eine hohe Bedeutung zukam. In beiden Fällen konnte das Anfangsbeispiel recht problemlos nachkonstruiert werden. Dennoch fiel bei der Testperson mit geringen Vorkenntnissen deutlich auf, dass die Hauptproblematik in der Benutzung des CMS an sich lag, und erst eine gewisse Einarbeitungszeit in die Erstellung gewöhnlicher Drupal Inhalte nötig war. Hingegen fiel es der Testperson mit gewissen Vorkenntnissen deutlich leichter, mit der Handhabung zurecht zu kommen.

Danach erfolgte das Experimentieren mit eigenen Abfragen klar ersichtlich stark geprägt von der verfügbaren Hilfestellung durch Anwendungsbeispiele. Es wurde stets versucht ähnliche Abfragen zu erzeugen und beide Testpersonen wagten sich nur zögerlich bzw. gar nicht an SPARQL Prefixes, die nicht in den Beispielen verwendet wurden. Positiv herauszustreichen ist, dass beiden TestprobandInnen dennoch recht rasch eigene Seiten mit sinnvollen Rückgaben gelangen. Die Benutzung war jedoch von hoher Experimentierbereitschaft und Fehlschlägen geprägt.

Durch anschließende Gespräche konnten einige Erfahrungen gewonnen werden:

- Es wurde empfunden, dass sich das Modul sehr gut im Drupal System einfügt. Es bereitete keine Probleme sich von der Erstellung eines Drupal Standard Node Typs auf den neuen Node Typ umzustellen. Auch der Benutzungsfluss mit dem Formular und dem *Generate* Button schien klar ersichtlich.
- Deutlich problematischer wurde die große Zahl an unterschiedlichen Eigenschaftstypen und Prefixes betrachtet. Wahrscheinlich wichen deshalb beide Testpersonen nur gering von den Prefixes der Anwendungsbeispiele ab.
- Eine Datenbank die die verfügbaren Eigenschaften bei der Eingabe enthält und vervollständigt wäre durchaus wünschenswert. Dies wäre jedoch mit einem erheblichen Mehraufwand bei der Installation verbunden, da eine solche Datenbank eine enorme Speichergröße voraussetzen würde.
- Die Anwendung des öffentlichen Zugangspunktes von DBpedia selbst bereitete die selben Probleme wie die Benutzung des Moduls. Auch hier lag die Problematik in der Auswahl der geeigneten Prefixes und Klassifizierungsschemen.

Die Programmtests zeigten sehr deutlich, dass die Verwendung von SPARQL Abfragen zur Inhaltsintegration nur mit guter Hilfestellung oder guten Vorkenntnissen funktionieren. Eine hohe Bereitschaft zum Erlernen des Musters von Abfragen ist in jedem Fall notwendig. Hierbei spielt auch die uneinheitliche Linie bei den DBpedia Klassifizierungen eine Rolle. Dies macht zusätzliche unterstützende Maßnahmen bei der Auswahl geeigneter Eigenschaften sinnvoll, beispielsweise durch eine Autovervollständigung bei Formulareingaben. Auf der anderen Seite würde eine einheitliche Klassifizierung in DBpedia die Anwendung erheblich erleichtern, auch wenn eine solche wahrscheinlich nur mit hohem manuellen Aufwand bei der Datenaufbereitung in der DBpedia Datenbank erzielt werden könnte.

Die Anwendung des Moduls würde sich praktisch daher nur bezahlt machen, wenn eine hohe Zahl an Inhalten erstellt werden müsste. In Szenarien, bei denen eine komplette Datenbank manuell aufgebaut werden müsste, scheint der Einsatz eines semantischen Moduls durchaus sinnvoll zu sein. In jedem Fall sollte für eine praktische Anwendung die an unerfahrene BenutzerInnen gerichtete Unterstützung noch ausgebaut werden.

5.5.3 Feedback und Diskussion

Für externe Kritiken und Ansichten wurde das Modul zusätzlich im Entwicklungsforum von Drupal vorgestellt. Zu diesem Zweck entstand eine frei verfügbare Testinstallation

(vgl. 5.4.1). Dieser Versuch brachte jedoch nur wenige kurze Rückmeldungen, die zwar als aufmunternd gewertet werden können, jedoch nur eine geringe Bereitschaft zeigen, sich mit dem Modul tiefer auseinander zu setzen. Dies lässt sich womöglich auf die Tatsache zurückführen, dass der Fokus der letztlich unkommerziellen und öffentlichen Gemeinschaft auf der Entwicklung des eigenen Systems liegt. Auch wenn die Unterstützung bei Problemen und Fragen mit dem System bewundernswert war, schien sie leider auch dort zu enden. Daher konnten aus den Programmtests mit den beiden Testpersonen einige Erfahrungen mehr gewonnen werden, die auf das Potential und die Schwächen des Moduls hinweisen.

Das vorliegende Endprodukt befindet sich insgesamt in einer fortgeschrittenen Phase, der Entwicklungsprozess soll aber bewusst nicht als abgeschlossen betrachtet werden. Die zugrundeliegenden Programmierschnittstellen, die SPARQL und RDF APIs befinden sich selbst noch in einer fortschreitenden Entwicklung, was einige Änderungen für die Zukunft offen lässt. Somit entspricht es dem Charakter des Drupal Projektes, wenn ein Modul nur eine Momentaufnahme eines fortlaufenden Prozesses darstellt, und mit dem gesamten System weiterwächst. Durch die zahlreichen Arbeiten in Richtung externer und interner semantischer Schnittstellen befindet sich das Drupal System auf einem vielversprechenden Weg zu einer der ersten semantischen Plattformen mit praktischem Nutzen. Die vorliegenden Technologien machen schon heute erste Anwendungen möglich, wie im praktischen Teil dieser Arbeit gezeigt wurde. Es ist lediglich eine Frage der Zeit bis weitere Module und Anwendungen folgen werden.

6 Abschluss

Im letzten Kapitel werden die wichtigsten Ergebnisse besprochen und die Arbeit wird abschließend zusammengefasst. Daneben enthält dieser Teil einen kritischen Rückblick auf den praktischen Teil der Arbeit aus Entwicklungssicht und einen Ausblick auf die nahe Zukunft des Semantic Web Content Management.

6.1 Zusammenfassung

Semantische Beschreibung von Daten können einen neuen und vielseitigeren Zugang zu Information im Internet ermöglichen. Anstatt Seiten nach Schlagworten zu durchsuchen, können Daten nach ihrer tatsächlichen Bedeutung gefunden, und Schlüsse zwischen ihnen gezogen werden. Die Organisation von Daten durch Content Management Systeme ist ein breites Anwendungsfeld, in dem semantische Beschreibungen zum Einsatz kommen können. Dies betrifft nicht nur die Synchronisation zwischen Systemen und den damit verbundenen dezentralen Zugriff auf Information, sondern auch den redaktionellen Prozess selbst. Semantische Suchabfragen können dazu dienen, die Integration von Inhalten zu beschleunigen und RedakteurInnen somit zu entlasten.

In dieser Arbeit wurden die wichtigsten Konzepte des Semantic Web und der zugrundeliegenden Technologien besprochen. Es wurde auf Internet Auszeichnungssprachen, Vokabularsprachen und Ontologien eingegangen. Ebenso wurden die Abfragesprache SPARQL und Anwendungen des Semantic Web vorgestellt. Danach wurden die Grundzüge und Aufgaben eines Content Management Systems definiert. Das Potential und die Anwendungsszenarien eines semantischen Content Managements wurden anhand von Beispielen aufgezeigt und ein Querschnitt durch die wichtigsten freien Systeme gezogen, um die Verfügbarkeit semantischer Dienste zu bewerten. Aus diesem Grund wurden Forschungsfragen definiert, die nach einem Einblick in die System ausgewertet wurden.

Im zweiten Teil der Arbeit wurde die semiautomatische Inhaltserstellung durch semantische Abfragen in einer Beispielanwendung realisiert. Zu diesem Zweck wurde eine

Schnittstelle im Content Management System Drupal implementiert, die einen Zugriff auf semantische Daten der Datenbank DBpedia ermöglicht. Zunächst wurde auf die Konzepte der einzelnen Systeme eingegangen und die Entwicklung in Drupal beleuchtet. Ebenso wurde auf die vorhandenen APIs eingegangen, auf die die Anwendung aufbaut. Anschließend wurde die Umsetzung anhand eines zyklischen Vorgehensmodells beschrieben. Es wurden Anforderungen und Szenarien definiert und ein Entwurf erstellt. Danach folgte eine Darstellung der Implementierung und eine Auswertung im Rahmen von Programmtests.

6.2 Diskussion

Das Semantic Web ist ein sehr aktives Forschungsgebiet, das schrittweise Einzug in das Content Management im Internet hält. Anwendungen wie Piggy Bank[1] zeigen eine völlig neue Form der Organisation von Inhalten und lassen auf das große Potential schließen, das semantische Dienste in Zukunft in diesem Bereich haben könnten. In Content Management Systemen könnte das Semantic Web etwa genutzt werden, um dezentral auf Information zuzugreifen, Inhalte zu Synchronisieren oder den redaktionellen Prozess durch semantisch erstellte Seiten zu beschleunigen. Auch aus BenutzerInnensicht könnte eine semantische Organisation von Daten zu neuartigen Zugängen bei der Erschließung von Inhalten und Formen der Auffindung von Daten führen.

In Open Source Systemen sind derartige Dienste heute jedoch nur ansatzweise verfügbar und das Ziel einer kompletten Vernetzung unterschiedlicher CMS scheint noch in weiter Ferne zu liegen. Externe Softwarelösungen, um Informationen auszutauschen, beginnen sich hingegen langsam zu etablieren. Die Projekte Triplify[2] und SIOC[3] bieten Schnittstellen an, die CMS über eine einheitliche Ontologie miteinander verknüpfen sollen. Eine Mehrzahl der großen CMS verfügt bereits über Schnittstellen zu diesen Lösungen. Durch die Unterstützung des W3C stehen diese Bestrebungen tatsächlich vor ihrer Standardisierung.

In wenigen Systemen wird auch an internen Lösungen und Schnittstellen gearbeitet. Solche Schnittstellen bieten den Vorteil, besser an ein bestimmtes System angepasst zu sein. Daher besteht der Anreiz systemeigener Entwicklungen in einer einfacheren Handhabung, die sich auch an unerfahrene BenutzerInnen richtet. Das Content Management

[1] http://simile.mit.edu/wiki/Piggy_Bank (Stand: 01.10.08)
[2] http://triplify.org (Stand: 01.10.08)
[3] http://sioc-project.org (Stand: 01.10.08)

System Drupal bietet solche Schnittstellen an, die erste praktische Anwendungen ermöglichen. Auch wenn solche Anwendungen bisher nur in begrenzter Zahl vorliegen, lassen sie erkennen, dass die Ziele des Semantic Web Content Management bereits heute umsetzbar sind. Daher ist es vielleicht nur noch ein kleiner Schritt bis sich diese Ideen tatsächlich etablieren.

6.2.1 Reflexion aus Entwicklungssicht

Aus Entwicklungssicht hat sich die Arbeit mit dem Drupal Framework als eher schwierig erwiesen. Das Content Management System bietet sehr viele Funktionen und Hilfsmittel, die die Entwicklung erleichtern sollen. Vorgegebene Datenstrukturen schränken die Freiheiten aber auch ein. So war es für das Projekt hinderlich, dass Formulare nicht frei definiert werden konnten. Dies führte dazu, dass der Zugriff auf Formularfelder durch das externe Skript nicht direkt möglich war. Beispielsweise kann in Drupal zwar ein HTML Button erstellt werden, das System geht aber automatisch davon aus, dass der Button die Methode *submit* erhält (siehe hierzu [Drupal, 2008b]). Die Definition einer anderen Methode war unmöglich, weshalb immer wieder Eingriffe in den Quelltext nötig waren. Insgesamt lässt sich sagen, dass das System zahlreiche Vorgaben mitbringt, die zwar AnfängerInnen unterstützen sollen, bei manchen Szenarien jedoch wie im Fall des unflexiblen Typs eines HTML Buttons am Ziel vorbeiführen.

Das CMS Drupal befindet sich in stetiger Entwicklung und an der Lösung vieler dieser Probleme wird bereits gearbeitet. Die vorliegende Version 6.2 wurde bereits während der Arbeit durch die kompatible Version 6.4[4] ersetzt und es ist bereits eine Veröffentlichung von Version 7[5] geplant. Auch die Arbeit mit der SPARQL API gestaltete sich nicht problemlos, da sich das Projekt noch in einer frühen Phase befand. Anfänglich wurden einige an sich gültige Abfragen von der API zurückgewiesen, weshalb der Algorithmus angepasst werden musste. Das bestehende Ergebnis lässt gewiss Raum für Verbesserungen offen, da jedoch auch noch intensiv an den zugrundliegenden RDF und SPARQL Modulen gearbeitet wird, ist ohnehin davon auszugehen, dass weitere Anpassungen nötig werden könnten.

Positiv herauszustreichen ist die breite Unterstützung des Projektes durch eine Community, und die zahlreichen angebotenen Hilfeforen und Dokumentationen. Die Plattform bietet sogar eigene CVS Konten an, um bei der Entwicklung von Modulen zu helfen.

[4] http://drupal.org/ (Stand: 15.09.08)
[5] http://drupal.org/contributors-guide (Stand: 15.09.08)

Dadurch können andere EntwicklerInnen beispielsweise bereits im Entstehungsprozess Feedback zu einer Arbeit geben und etwaige Fehler melden. Betrachtet man den völlig offenen sozialen Charakter der Plattform, ist es erstaunlich, dass ein derart stabiles und geradliniges CMS entstehen konnte, das sich durch eben diesen Ansatz in einer stetigen Entwicklung befindet. Somit ist es schwierig vorherzusehen, in welche Richtung das System gehen, und welche Rolle Semantik darin spielen wird. An den bestehenden Ansätzen sieht man jedenfalls, dass in diesem Bereich einiges möglich ist.

6.3 Ausblick

Das Semantic Web Content Management ist ein Begriff, der sich erst langsam von einem theoretischen Ansatz zu praktischen Anwendungen hin löst. Dennoch ist an den vielen Projektlösungen ein Bestreben zu erkennen, Content Management Systeme an das Semantic Web heranzuführen. Schnittstellen wie die RDF[6] und SPARQL APIs[7] von Drupal lassen einige Entwicklungen für die Zukunft erwarten. Da sich auch diese Projekte noch in einer frühen Entwicklungsphase befinden, lässt sich aber nur schwer vorhersagen, welche Rolle sie in der Zukunft tatsächlich spielen werden.

Die Entwicklung des Semantic Content Moduls zeigt deutlich auf, dass Anwendungsfelder wie die semiautomatisierte Erstellung von semantischen Inhalten bereits möglich und realisierbar sind. Die Weiche zu einem semantischen Content Management ist also bereits gestellt und die Umsetzung semantischer Anwendungen ist ohne großer Entwicklungsarbeit möglich. Offen bleibt jedoch, inwiefern diese Möglichkeiten in naher Zukunft angenommen werden und was daraus entsteht wird.

[6] http://drupal.org/project/rdf (Stand: 01.10.08)
[7] http://drupal.org/project/sparql (Stand: 01.10.08)

7 Danksagung

Ich möchte mich im Rahmen dieser Arbeit bei allen Menschen bedanken, die mir bei der Bewerkstelligung dieses Projektes zur Seite gestanden sind. In erster Linie möchte ich mich bei meiner Betreuerin Monika Lanzenberger für die zahlreichen Ratschläge und Bemühungen bedanken. Ich möchte mich bei meiner Familie für die Unterstützung bedanken, ohne die ich mein Studium nicht hätte absolvieren können. Mein Dank gilt auch allen Kollegen und Kolleginnen, die mir stets zur Seite gestanden sind und bei der Verwirklichung dieses Projekts geholfen haben. Vielen Dank an euch alle!

8 Literaturverzeichnis

[Antoniou, 2004] Antoniou, Grigoris; Harmelen Frank van. 2004. *A Semantic Web Primer*. MIT Press. ISBN: 978-0262012102.

[Auer, 2007] Auer, Sören; Bizer Christian; Lehmann Jens; Kobilarov Georgi; Cyganiak Richard; Ives Zachary. 2007. DBpedia: A Nucleus for a Web of Open Data. *Aberer et al. (Eds.): The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference*. ISBN: 9783-540762973.

[Beck, 2004] Beck, Norman; Auer Sören. 2004. *Semantic Web Content Management*. Fähnrich, K.-P. ISBN: 3-898380-57-2.

[Berners-Lee, 2001] Berners-Lee, Tim. 2001. Semantic Web. *Scientific American 5/2001*. http://www.sciam.com/article.cfm?id=the-semantic-web (Stand: 12.10.08).

[Boehm, 1988] Boehm, Barry. 1988. A Spiral Model of Software Development and Enhancement. *IEEE Computer. Vol. 21.*

[Bray, 2006] Bray, Tim. 2006. *Extensible Markup Language (XML) 1.0, 4th Edition*. W3C. http://www.w3.org/TR/REC-xml/ (Stand: 12.10.08).

[Brickley, 2004] Brickley, Dan. 2004. *RDF Vocabulary Description Language 1.0: RDF Schema*. W3C. http://www.w3.org/TR/rdf-schema/ (Stand: 12.10.08).

[DBpedia, 2008] DBpedia. 2008. *Interlinking DBpedia with other Data Sets*. DBpedia Open Community. http://wiki.dbpedia.org/Interlinking (Stand: 28.07.08).

[Diestelkamp, 2005] Diestelkamp, Eike; Birkenhake Benjamin. 2005. *Die Semantic Web Ernüchterung oder wie ein CMS Semantik lernt*. http://t3n.yeebase.com/magazin/ausgaben/artikel/die-semantic-web-ernuechterung/ (Stand: 12.10.08).

[Doyle, 2004] Doyle, Bob. 2004. *CMS Genesis - Who did What When?* http://www.econtentmag.com/Articles/ArticleReader.aspx?ArticleID=6819 (Stand: 12.10.08).

[Drupal, 2006] Drupal. 2006. *History*. Drupal Association. http://drupal.org/node/769 (Stand: 12.10.08).

[Drupal, 2008a] Drupal. 2008. *Drupal API Reference*. Drupal Association. http://api.drupal.org/ (Stand: 06.07.08).

[Drupal, 2008b] Drupal. 2008. *Drupal Development Issues*. Drupal Association. http://drupal.org/node/72855 (Stand: 15.09.08).

[Drupal, 2008c] Drupal. 2008. *Drupal General concepts*. Drupal Association. http://drupal.org/node/19828 (Stand: 06.07.08).

[Drupal, 2008d] Drupal. 2008. *Drupal Node Types*. Drupal Association. http://drupal.org/node/21947 (Stand: 22.07.08).

[Drupal, 2008e] Drupal. 2008. *Drupal RDF API*. Drupal Association. http://drupal.org/handbook/modules/rdf (Stand: 22.07.08).

[Drupal, 2008f] Drupal. 2008. *Drupal RDF Modules*. Drupal Association. http://drupal.org/project/Modules/category/116 (Stand: 22.07.08).

[Gilbane, 2000] Gilbane, Frank. 2000. What Is Content Management? *The Gilbane Report 8/2000*. http://gilbane.com/artpdf/GR8.8.pdf (Stand: 12.10.08).

[Graf, 2006] Graf, Helgen. 2006. *Drupal. Community-Websites entwickeln und verwalten mit dem Open Source-CMS*. Addison-Wesley. ISBN: 3827323215.

[Kerres, 2006] Kerres, Michael. 2006. *Potenziale von Web 2.0 nutzen*. Andreas Hohenstein und Karl Wilbers. http://mediendidaktik.uni-duisburg-essen.de/system/files/web20-a.pdf (Stand: 12.10.08).

[Kim, 2005] Kim, Jaehong; Jang Minus; Ha Young-Guk; Sohn Joo-Chan; Lee Sang-Jo. 2005. *MoA: OWL Ontology Merging and Alignment Tool for the Semantic Web*. Springer Berlin. ISBN: 978-3-540-26551-1.

[Klein, 2001] Klein, Michel. 2001. XML, RDF, and Relatives. *Intelligent Systems 16/2001, IEEE*.

[Koller, 2005] Koller, Andreas; Pellegrini Tassilo. 2005. Content Managament Systeme auf dem Weg zum Semantic Web. *content management magazin 2/2005*. http://www.community-of-knowledge.de/pdf/CMS_SemanticWeb.pdf (Stand: 12.10.08).

[Kristöfl, 2003] Kristöfl, Robert. 2003. *Evaluation von Content Management Systemen*. BMBWK. http://www.plan2net.at/fileadmin/p2net/downloads/typo3edu/bmbwk-Evaluation_CMS.pdf (Stand: 12.10.08).

[MAMBO, 2006] MAMBO. 2006. *A brief History of the Mambo Open Source Project*. MAMBO Foundation. http://mambo-foundation.org/content/view/21/2/ (Stand: 12.10.08).

[Manola, 2004] Manola, Frank; Miller Erik. 2004. *RDF Primer*. W3C. http://www.w3.org/TR/REC-rdf-syntax/ (Stand: 12.10.08).

[McGuiness, 2004] McGuiness, Deborah L.; Harmelen Frank van. 2004. *OWL Web Ontology Language Overview*. W3C. http://www.w3.org/TR/owl-features/ (Stand: 12.10.08).

[Noy, 1999] Noy, Natalya F.; Musen Mark A. 1999. SMART: Automated Support for Ontology Merging and Alignment. *Twelfth Workshop on Knowledge Acquisition, Modeling, and Management*. http://citeseer.ist.psu.edu/article/noy99smart.html (Stand: 12.10.08).

[Noy, 2000] Noy, Natalya F.; Musen Mark A. 2000. PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. *Proceedings of the Seventeenth National Conference on Artificial Intelligence, AAAI Press*. ISBN: 0-262-51112-6.

[Pellegrini, 2006] Pellegrini, Tassilo; Blumauer Andreas. 2006. *Semantic Web. Wege zur vernetzten Wissensgesellschaft*. 1 edn. Springer Verlag Berlin. ISBN: 978-3540293248.

[Prud'hommeaux, 2008] Prud'hommeaux, Eric; Seaborne Andy. 2008. *SPARQL Query Language for RDF*. W3C. http://www.w3.org/TR/rdf-sparql-query/ (Stand: 12.10.08).

[Raggett, 1999] Raggett, Dave; Le Hors Arnaud. 1999. *HTML 4.01 Specification*. W3C. http://www.w3.org/TR/REC-html40/ (Stand: 12.10.08).

[Roggenbuck, 2006] Roggenbuck, Robert; Sperber Wolfram. 2006. CMS für das Semantic Web: Eine Analyse anhand der Erfahrungen des Math und Industry-Projekts. *In: XML Tage 2006 in Berlin - Tagungsband*. ISBN: 3-938863-76-5.

[Typo3, 2006] Typo3. 2006. *The History of Typo3*. Typo3 Association. http://typo3.com/History.1268.0.html (Stand: 12.10.08).

[Zschau, 1999] Zschau, Oliver. 1999. *Kennzeichen von Content Management Systemen.* Feig und Partner. http://www.contentmanager.de/magazin/artikel%2_kennzeichen_von_content_management_systemen.html (Stand: 12.10.08).

[Zschau, 2001] Zschau, Oliver; Traub Dennis; Zahradka Rik. 2001. *Web Content Management - Websites professionell planen und betreiben.* 2 edn. Galileo Press. ISBN: 978-3898421577.

Abbildungsverzeichnis

2.1 Ein gerichteter Graf für 3 RDF Aussagen. 15
2.2 Freebase (http://www.freebase.com) 21
2.3 DBpedia (http://www.dbpedia.org) 21
2.4 PowerSet (http://www.powerset.com) 22
2.5 AdaptiveBlue (http://www.adaptiveblue.com) 23

3.1 Das Schema eines Web Content Management Systems [Kristöfl, 2003]. . . 28
3.2 Umsetzung von Labels in ein RSS Schema [Koller, 2005]. 33
3.3 TYPO3 (http://www.typo3.org) 40
3.4 Joomla! (http://www.joomla.org) 42
3.5 Drupal (http://www.drupal.org) 43
3.6 MODx (http://www.modx.com) 45
3.7 WordPress (http://www.wordpress.org) 47

4.1 Blöcke und Boxen in Drupal 54
4.2 Ein Beispielmodul im Administrationsbereich Drupal 56
4.3 Anzeige eines Beispielmoduls im Frontend von Drupal 59
4.4 Schnittstellen zwischen semantischen Plattformen [DBpedia, 2008] 62

5.1 Vorgehensmodell auf Basis von [Boehm, 1988] 68
5.2 Semantic Content Seite mit ausgeklappten Prefixes. 73
5.3 Mit dem Semantic Content Modul erstellte Seite 74
5.4 Semantische Abfrage im Semantic Content Modul 75
5.5 Dynamisch erstellte Ergebnisseite 76
5.6 Programmarchitektur und Positionierung im Drupal System 77
5.7 Dateistruktur des Moduls 78
5.8 Mock-up der Gesamtansicht 81
5.9 Mock-up des Query Builders 81
5.10 Ansicht des implementierten Formulars 86
5.11 Kommunikationsschema des Semantic Content Moduls 87

Listings

2.1 HTML Code Beispiel 12
2.2 XML Code Beispiel 12
2.3 RDF Aussagen 14
2.4 SPARQL Query Beispiel 19

4.1 Info Datei eines Drupal Moduls 56
4.2 Auszug aus der Module Datei eines Drupal Moduls 57
4.5 SPARQL Query in DBpedia 63
4.6 SPARQL API Query Beispiel 65

5.1 Einfache textuelle SPARQL Abfrage 73
5.2 SPARQL Abfrage durch das Semantic Content Modul 74
5.3 Verbesserte SPARQL Abfrage aus 5.1 74
5.4 Definition der Zugriffsrechte im Semantic Content Modul 82
5.5 Implementierung der Zugriffsrechte 83
5.6 Definition des Node Typs im Semantic Content Modul 84
5.7 Implementierung des Formulars des Semantic Content Moduls 84
5.8 Ausgabefunktion des Semantic Content Moduls 87

9.1 Vollständige Implementierung eines Drupal Moduls 107

9 Anhänge

9.1 Anhang A: Hooks in Drupal

Dieser Anhang beinhaltet eine Auflistung der Hooks in der Drupal Grundintallation. Siehe hierzu Abschnitt 4.2.3

- hook_access: Definiert die Zugangseinschränkungen.
- hook_actions_delete: Führt Code aus nachdem *delte action* aufgerufen wurde.
- hook_block: Deklariert Blöcke.
- hook_boot: Setup Aufgaben.
- hook_comment: Verhalten bei Kommentaren.
- hook_cron: Führt periodische Aktionen durch.
- hook_db_rewrite_sql: Dient zur Umschreibung von SQL Abfragen, meist zur Zugriffskontrolle.
- hook_delete: Reagiert auf einen Löschvorgang einer Node.
- hook_disable: Führt Aktionen durch wenn ein Modul deaktiviert wird.
- hook_elements: Hierüber können Module eigene Form API Elemente und deren Grundwerte festlegen.
- hook_enable: Führt Aktionen durch wenn ein Modul aktiviert wird.
- hook_exit: Führt Aufgaben nach dem Ausstieg aus.
- hook_file_download: Kontolliert den Zugriff zu privaten Downloads und spezifiziert HTTP Headers.
- hook_filter: Definiert Inhaltsfilter.
- hook_filter_tips: Bietet Tips zur Benutzung diverser Filter an.
- hook_flush_caches: Fügt eine Liste von Tabellen ein, die bereinigt werden sollen.
- hook_footer: Fügt HTML Code am Dateiende ein.
- hook_form: Zeigt eine Form zur Bearbeitung von Nodes an.
- hook_form_alter: Führt Änderungen durch bevor eine Form gerendert wird.
- hook_get_parents: Findet alle übergeordneten Elemente einer ID.
- hook_get_tree: Erzeugt eine hierarchische Repräsentation eines Vokabulars.

- hook_get_vocabularies: Gibt ein Array mit dem gesamten Vokabular zurück.
- hook_help: Bietet Online Hilfe an.
- hook_hook_info: Erzeugt eine Liste von Triggers (Events) die in dem jeweiligen Modul von BenutzerInnen verwendet werden dürfen.
- hook_init: Führt Setup Aufgaben aus.
- hook_insert: Antwortet auf das Einfügen einer Node.
- hook_install: Installiert das aktuelle Datenbankschema.
- hook_link: Definiert interne Drupal Links.
- hook_link_alter: Führt Änderungen durch, bevor Links gerendert werden. Hierdurch können Links von anderen Modulen integriert oder entfernt werden.
- hook_load: Ladete spezifische Informationen zu einem Node Typ.
- hook_locale: Erlaubt Modulen, eigene Text Gruppen zu definieren, in die übersetzt werden kann.
- hook_mail: Bereitet eine Nachricht auf Basis von Parametern vor.
- hook_mail_alter: Ändert die Form wie emails von Drupal versedet werden.
- hook_menu: Definiert Menüelemente.
- hook_menu_alter: Ändert Daten die in der Tabelle (menu_router) abgelegt werden, nachdem hook_menu aufgerufen wurde.
- hook_menu_link_alter: Ändert Daten die in der Tabelle (menu_links) abgelegt werden, nachdem menu_link_save() aufgerufen wurde.
- hook_nodeapi: Reagiert auf Nodes die von anderen Modulen definiert wurden.
- hook_node_access_records: Setzt die Rechte für Nodes die in die Datenbank schreiben.
- hook_node_grants: Informiert das Node Zugriffssystem welche Rechte ein User besitzt.
- hook_node_info: Definiert Node Typen die von Modulen bereitgestellt werden.
- hook_node_operations: Fügt Massenoperationen für Nodes hinzu.
- hook_node_type: Reagiert auf Änderunegn der Node Typen.
- hook_perm: Definiert Benutzerrechte.
- hook_prepare: Wird aufgerufen nachdem ein Modul geladen, aber bevor es angezeigt wird.
- hook_profile_alter: Führt Änderungen an Elementen des Profils durch bevor sie gerendert werden.
- hook_requirements: Überprüft die Installationsanforderungen und führt den Statusreport durch.
- hook_schema: Definiert die aktuelle Version des Datenbankschemas.
- hook_search: Definiert eine eigene Routine für Suchen.
- hook_search_preprocess: Bereitet Text für den Such Index auf.

- hook_taxonomy: Reagiert auf Änderungen der Taxonomie.
- hook_term_path: Erlaubt Modulen, einen alternativen Pfad zu den Teilen die es verwaltet bereitzustellen.
- hook_theme: Registriert ein Tehme.
- hook_theme_registry_alter: Ändert die Registrierungsinformation die von hook_theme() zurückgegeben wird.
- hook_translated_menu_link_alter: Ändert einen Link des Menüs nachdem er Übersetzt, aber bevor er gerendert wird.
- hook_uninstall: Entfernt alle Tabellen und Variablen die von einem Modul gesetzt wurden.
- hook_update: Antwortet auf ein Update einer Node.
- hook_update_index: Führt ein Update des gesamten Index eines Moduls durch.
- hook_update_N: Führt eine einfaches Update aus.
- hook_user: Reagiert auf Aktionen eines Benutzerkontos.
- hook_user_operations: Führt Massenoperationen auf BenutzerInnen aus.
- hook_validate: Verifiziert eine Form zur Bearbeitung von Nodes.
- hook_view: Zeigt eine Node an.
- module_hook: Ermittelt ob ein Modul einen Hook implementiert oder nicht.
- module_implements: Ermittelt welche Module einen Hook implementieren.
- module_invoke: Ruft einen Hook in einem bestimmten Modul auf.
- module_invoke_all: Ruft einen Hook in allen freigeschalteten Modulen auf, die diesen Hook implementieren.

9.2 Anhang B: birthday.module Datei

Listing 9.1: birthday.module

```
<?php
//; $ Id: birthday.module, Semantic Web Content Management $

/**
* @file
* Beschreibung
*/

// Zugriffsrechte definieren
function birthday_perm() {
  return array('access birthday content');
}

// Modul als Block definieren
function birthday_block($op='list', $delta=0) {
  if ($op == "list") {
     $block[0]["info"] = t('Birthday Block');
     return $block;
  } else if ($op == 'view') {

  // Inhalt des Blocks
    $block_content = '<p>Geburtstagskinder:</p>';

    $query="SELECT u.name, u.uid from profile_values p, users u
            WHERE p.fid = 1 AND p.uid = u.uid AND p.value like '%";
  $query.=date("md");
  $query.="%';";

    $result = db_query($query); // User Namen extrahieren
  while ($users = db_fetch_object($result)) {
      $block_content .= l($users->name, $users->uid) . '<br />';
    }

  // Set Up des Blocks
    $block['subject'] = 'Birthday Block';
    $block['content'] = $block_content;
    return \$block;
  }
}
```